

Black Swan Events and Machine Learning

by

Lamia Tabassum

16101203

Prasenjit Das

16201107

Tasneem Bin Ahmed

16101312

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

Department of Computer Science and Engineering
Brac University
January 2021

© 2021. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Lamia Tabassum

Lamia Tabassum
16101203



Prasenjit Das
16201107

Tasneem Bin Ahmed

Tasneem Bin Ahmed
16101312

Approval

The thesis titled “Black Swan Events and Machine Learning” submitted by

1. Lamia Tabassum (16101203)
2. Prasenjit Das (16201107)
3. Tasneem Bin Ahmed (16101312)

has been accepted as satisfactory in partial fulfilment of requirement for the degree of Bachelor of Science in Computer Science on January 10, 2021 at Fall 2020 semester.

Examining Committee:

Supervisor:
(Member)



Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
BRAC University


Dr. Mahbubul Alam Majumdar
Professor and Dean
School of Data and Sciences
BRAC University

Program Coordinator:
(Member)



Dr. Md. Golam Rabiul Alam
Associate Professor
Dept. of Computer Science & Engineering
School of Data and Sciences
BRAC University

Head of Department: (Chair)



Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
BRAC University

Dr. Mahbubul Alam Majumdar
Professor and Dean
School of Data and Sciences
BRAC University

Abstract

Black Swan events refer to hard-to-predict and rare events that have a low probability of occurrence but have widespread impacts whenever they occur, be it positive or negative. These events can either be in the form of natural disasters, political events or even as catastrophic financial market crashes. From the stock market crash in 1987 to the global financial crisis in 2008 and to the most recent COVID-19 stock market crash in 2020 that has devastated the world economy, Black Swan events in finance have severe consequences on global socio-economy due to which not contemplating them for their rarity is no longer an option. The purpose of this research study is to use machine learning tools and statistical tools in order to detect, fit and predict such critically catastrophic financial market crashes in anticipation of capturing black swan events in future. For this, we have used the concepts of both crashes as “outliers” [4] as termed by the authors Didier Sornette and Anders Johansen and also crashes characterized as “drawdowns” [4] by author Emilie Jacobsson as guidance and then analyzed and fitted major financial market crashes using different statistical tools including an advanced non-linear generalized log-periodic power law model proposed by Sornette and Johansen, and also different machine learning tools including neural networks in order to predict the trend of financial market indices. Due to complex time series of financial market index, we found thresholds for different financial crash data exhibiting different behavior, based on which we detected historical financial market crashes using statistical and machine learning tools including neural networks. In order to find thresholds, we have considered the viewpoints of author Emilie Jacobsson and authors Didier Sornette and Anders Johansen. Using data from six major global financial market indices, we also cross validated our models in order to evaluate them. Using these thresholds, any supervised machine learning model can learn about financial market crashes.

Keywords: Black Swan events; Log Periodic Power Law; LPPL; Machine Learning; Prediction; Crashes; Financial Market Crash-trends; Decision tree; Linear Regression; Logistic regression; LSTM; Stateful LSTM; Stateless LSTM; Support Vector Machine; Back Propagation; Adam Optimization Algorithm; k-fold Cross Validation; F-score; F_β score

Acknowledgement

First of all, all praise goes to the Great Almighty for whom our thesis has been completed without any major interruptions.

Secondly, we would like to express our utmost gratitude to our advisor and supervisor Dr. Mahbubul Alam Majumder Sir for his kindest support, valuable guidance and endless patience and trust in us during our research study. We are extremely thankful to Sir for introducing us to this research field and for motivating and encouraging us to pursue this research study.

Finally, we are very grateful to our family for their endless love and support without which it may not have been possible to continue our research. With their kind support and prayers, we are now on the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
Nomenclature	viii
1 Introduction	1
2 Related Work	3
3 Data Collection and Feature Selection	5
3.1 Data Preliminary Analysis	5
3.2 Data exploration	7
4 Model Description	15
4.1 Log Periodic Power Law	15
4.2 RNN LSTM (Statefull and Stateless)	16
4.3 Decision Tree	17
4.4 Linear Regression	19
4.5 Logistic Regression	20
4.6 Back Propagation	21
4.7 Support Vector machine	27
4.8 Adam Optimizer	32
4.9 K-Fold Cross Validation	33
4.10 F-score	33
5 Result Analysis	35
5.1 Log Periodic Power Law	35
5.2 RNN LSTM	36
5.2.1 Stateful & Stateless LSTM	36
5.3 Back Propagation	40
5.4 Regression	40

5.4.1	Linear & Logistic	40
5.5	Decision Tree	43
5.6	Support Vector Machine	46
5.7	Result Overview	48
6	Conclusion and Motivation for Future work	49
	Bibliography	51

List of Figures

3.1	Oct, 2008 market crash. The figure shows the two major index of US stock market DJI and S&P 500. Data on the black boxes are from DJI.	5
3.2	Shows the pick point of DJI before starting 2020 market crash.	6
3.3	Shows the pick point of DJI before starting Oct, 2008 market crash.	6
3.4	Correlation of daily returns of all eleven datasets	7
3.5	Least correlated datasets	7
3.6	Time series plot distribution of daily price returns over the last years.	8
3.7	Fluctuation of daily returns indicating volatility of the markets	8
3.8	Auto-correlation plot, suggest that daily return can not be a strong indicator for the price change in the next following days.	9
3.9	Distribution of daily returns. In SSE, there is fat tails indicating high volatility compared to others.	9
3.10	Frequency log-distribution of daily returns indicating extreme positive(green) and extreme negative(red) occurrence of return. Extreme negative returns less than -0.1 is more likely to contribute in a crash.	10
3.11	Duration of drawdowns.	10
3.12	Frequency log-distribution of daily returns indicating extreme drawdowns(>-1.5) occurrence of return. Such significant drawdowns have only happened two times in the S&P for nearly 70 years. These extreme drawdowns are more likely associated with crashes.	11
3.13	Drawdowns ranked from 1 to n.	11
3.14	Weibull distribution by rank.	12
4.1	Simple neural network	21
4.2	Basic structure of a neuron	21
4.3	calculation of weighted sum and activation of layer (L-1)	24
4.4	calculating weighted sum and activation of final layer (L). After that calculating the cost function	25
4.5	error signal from the layer (L) to the layer (L-1)	25
4.6	error signal from the layer (L-1) to the layer (L-2)	26
4.7	calculating weighted sum and activation of final layer (L). After that calculating the cost function	30
4.8	6-fold cross validation	33
5.1	LPPL fit on S&P 500 market index from year of 2008 up to 2010	35
5.2	: LPPL prediction on S&P 500 market index from year of 2008 to 2010	36
5.3	LPPL fit on S&P 500 market index from year of 2018 up to 2021	36
5.4	Finding the most accurate threshold for the test data set. (LSTM Stateless)	37

5.5	Finding the best best threshold for the test data set. (LSTM Stateful)	37
5.6	LSTM - Stateful	38
5.7	LSTM - Stateless	38
5.8	Test result: Stateful	38
5.9	Test result: Stateless	38
5.10	LSTM stateful test case 2004 to 2016	39
5.11	LSTM stateless test case 2004 to 2016	39
5.12	LSTM stateful test case during Covid-19	39
5.13	LSTM stateless test case during Covid-19	39
5.14	Finding the best best threshold for the test data set.(Linear regression)	40
5.15	Finding the best best threshold for the test data set.(Logistic regression)	41
5.16	Lin. Regression	41
5.17	Log. Regression	41
5.18	Test result: Linear	42
5.19	Test result:Logistic	42
5.20	Lin. regression test case 2004 to 2016	42
5.21	Log. regression test case 2004 to 2016	42
5.22	Lin. Regression test case during Covid-19	42
5.23	Log. Regression test case during Covid-19	42
5.24	Prediction: Linear	43
5.25	Prediction: Logistic	43
5.26	Finding best parameters	43
5.27	Decision tree model overview	44
5.28	Decision Tree test case from 2004 to 2016	45
5.29	Decision Tree test case during Covid-19	45
5.30	Support vecto machine model overview	46
5.31	Test result of SVM on S&P 500 data set	46
5.32	SVM from the year 2004 to year 2016	47
5.33	SVM test case during Covid-19	47

Chapter 1

Introduction

Following the major effects of the 2008 global economic crisis, the forecasting of the financial crisis has been one of the most common subjects in academic study over the last few years. Forecasting financial variables using advanced mathematical models has become one of the subjects most widely discussed in academic literature, industry, and other sectors. In the early days of financial market trading, it was almost impossible to predict financial market crashes. But through the growth and interconnection of the global socio-economy and due to technological advancements such as enhancement of machine learning techniques, many researchers have attempted to detect or model or to even predict these critical events using various approaches. Bubble in financial market is defined as a period of time that, through a sustained market acceleration, followed by a crash or a large decrease, goes from a very low to a large high. The existence of price bubbles in the financial market has a major influence on the market's success or failure. The two phases found in the market are Bull market which is when the financial market faces an upward trend in index price, and Bear market which is when the market faces a downward trend in index price. When the bear phase exceeds the norm limits, the markets crash. The main cause of such financial Black Swan events is due to the herding behaviour of financial market traders [15]. The possibility of all these events happening seems so unlikely as to be unintelligible through normal statistical methods, but for a term that was used to express impossibilities, ignoring makes them dangerous. Even with the regulators' insistence of low default rates, and enhanced safety measures for the global financial system, the risk is too great to simply dismiss the potential of another devastating economic crisis happening in the near future[14]. Hence, it is imperative to assume that these events have a possibility of occurring and plan accordingly using statistics or empirical data, especially in dire cases of predictions for markets and investments. The goal is to exploit the positive Black Swan Events while simultaneously preparing for the aftermath of its negative counterparts.

According to Sornette [5], "Financial Crashes are Outliers; the presence of outliers is a general phenomenon", large price losses are "outliers" that show the stock market's underlying properties. Proposed by Didier Sornette and Anders Johansen, one of the most successful approaches so far has been the "Log Periodic Power Law" model for detecting and capturing the trend of financial market indices. The main focus of our research study is to use machine learning tools and statistical tools to predict catastrophic stock market events based on the previous price pattern of market indices. We will backtest users' previous crash data of major financial

markets from different regions. For this, we will use the concept of viewpoints of author Emilie Jacobsson and authors Didier Sornette and Anders Johansen as guidance in order to analyze and fit major financial market crashes using different statistical tools including an advanced non-linear generalized log-periodic power law model proposed by Sornette and Johansen, and also different machine learning tools including neural networks in order to predict the trend of financial market indices. Initially, we have analyzed major stock market failures through different statistical methods. Then, we have gone through some working processes sequentially such as data collection and processing, features selection, fitting into model, k-fold cross validation, back testing, error measurement, comparing error with the output of other models. At the end of our research, we have been able to successfully capture and predict those failures using some machine learning algorithms as well as the Log-Periodic Power Law.

Chapter 2

Related Work

According to Taleb (2017), the Black Swan as an event that follows three criteria: 1. it is an outlier, since it lies far outside of the regular expectations, 2. It carries a catastrophic impact when it does occur, 3. Despite its outlier position, human behavior produces a desire to justify after the reality, making it explainable and predictable(Taleb, 2017)[15]. To capture more value from such events, Taleb’s advice is to focus on potential consequences of the unexpected by crediting all aspects rather than on the likelihood of the improbable happening, especially since making mistakes in the stock market can have devastating consequences. Chowdhury, R.A., Mahdy, M., Alam, T.N., & Quaderi, G.D. (2018)[21] described in that, The market collapse is directly attributed to the sudden immense rise in the volatility of 4 stocks. If the uncertainty tends to rise at a considerable pace, month after month, along with the increase in the stock price of companies on a border market such as the Dhaka stock exchange, then it must be known that the market is likely to crush or crash in the near future(Chowdhury & Mahdy, 2018).

Sornette. (2017) proposed in his book “Why Stock Markets Crash”[13] that, The root cause of the crash can be found in the intervening months and years, in the gradual build-up of industry co-operation, or in successful contacts between participants, often converted into an exponential increase in market prices. A crash happens when the economy has reached an uncertain stage and the volatility could have been caused by some minor disruption or operation. The fall is primarily due to the fragile situation.This book discusses the notion that a crash has an endogenous/internal cause in nature and that exogenous/external shocks function merely as key factors. As a result, the cause of the collapse is much more complicated than is often believed to be, as the economy as a whole is increasingly built as a self-organizing mechanism. In this way, the actual cause of a collapse may be considered structural uncertainty. Stock market crashes are often unforeseen for most people, even economists. There has never been a market crisis when the metrics look grim. In the opposite, asset markets and macro-economic flows (output, wages, etc.) tend to increase and strengthen until the crisis. This explains why most people, particularly economists, are taken completely by surprise by a crash. (The Great Crash of October 1929/Wall Street collapsed Black Tuesday, October 29, 1929). These features have paved the way to reexamining evidence of bubble, a “fad” or “herding” behavior by studying each stock returns. It is also compatible with the possibility that fad or crowd psychology could have played a role in the growth of the bubble, its decline, and following volatility. By local self-reinforcing

imitation between merchants, a crash can be induced. A crash is not a particular outcome of bubble, but can be specified by its risk intensity. In stock market prediction scenarios, artificial neural networks have proven to be an effective method for mapping complex and non-linear relationships. Yue-Gang Songa, Yu-Long Zhou, Ren-Jie Hanc (2018) at their paper “Neural networks for stock price prediction” [18] have analyzed and compared the predictive power of 5 NN models (BPNN, RBFNN, GRNN, SVMR, LS-SVMR) by making price prediction of 3 individual stocks (Bank of China, Vanke A, Kweichou Moutai). Criteria for performance adopted here had been the “mean-square error” (MSE) and the “mean average absolute percentage error” (MAPE)[9] also reviewed several forecasting models from different papers. Among all the reviewed model they found that the Black & Scholes option pricing model, time series volatility forecasting model, support vector machine does a better job. Chowdhury, R.A., Mahdy, M., Alam, T.N., & Quaderi, G.D. (2018)[21] also successfully implemented Black & Scholes model with some modification. From the above studies, it is very clear that catastrophic events also can be predicted with modern machine learning techniques like neural networks due its non-linearity and for the ability to handle large volume stock price data of financial market crashes.

Chapter 3

Data Collection and Feature Selection

3.1 Data Preliminary Analysis

As an illustrative example of the log-periodic signatures we want the log-periodic power law model to determine in stock indices during major financial market crash events worldwide, below are shown the case of one of the two major indices of the U.S stock market: DJI (Dow Jones Index) / DJIA (Dow Jones Index Average) :



Figure 3.1: Oct, 2008 market crash. The figure shows the two major index of US stock market DJI and S&P 500. Data on the black boxes are from DJI.

According to the figure above, In October, 2008, Dow Jones Industrial Average fell 77.68 points on one day trading because of rejection of the bailout bill by congress [20]. Eventually other major stock markets also faced a major fall.

At t_{max} , market price on its maximum before the crash happen and at t_{min} market price in its lowest position after the occurrence of crash. From the figure shown above, the market price achieves its maximum just before the incident of crash in April 2007 with a closing price of 13895.63 points and then index points plummeted till October, 2008 with a closing price of only 7808.92 points, and then price

rebounded. Now, in the following two box-plots shows the DJI index on a) before starting 2020 financial market crash and b) October, 2008 financial market crash



Figure 3.2: Shows the pick point of DJI before starting 2020 market crash.



Figure 3.3: Shows the pick point of DJI before starting Oct, 2008 market crash.

In the chart we can see a Bearish Engulfing pattern in September, 2007. That indicated that the market will move downward in near future. After that in October 2008, the market fell. Our goal is to find such candlestick patterns on the future market which will help us to point out the market fall[24].

3.2 Data exploration

In order to get the best result from our models, we have decided to consider the least correlated datasets among the eleven large financial market indices from different locations of the world. Which are Dow Jones Index(DJI, USA), S& P 500 (United States), Nasdaq (USA), Shanghai Stock Exchange (SSE), Swiss Market Index(SMI), BOVESPA Index (BVSP, Brazil), S&P BSE SENSEX (BSESN), India), Dax Index(Europe), Nikkei 225 (NKY, Japan), Hang Seng Index (HSI, Hong Kong), MXX(Mexico). Among these eleven indices, we have chosen the top six least correlated datasets which will be used to develop our models and predict major crashes. Seaborn heatmap is best to plot the correlation among datasets.

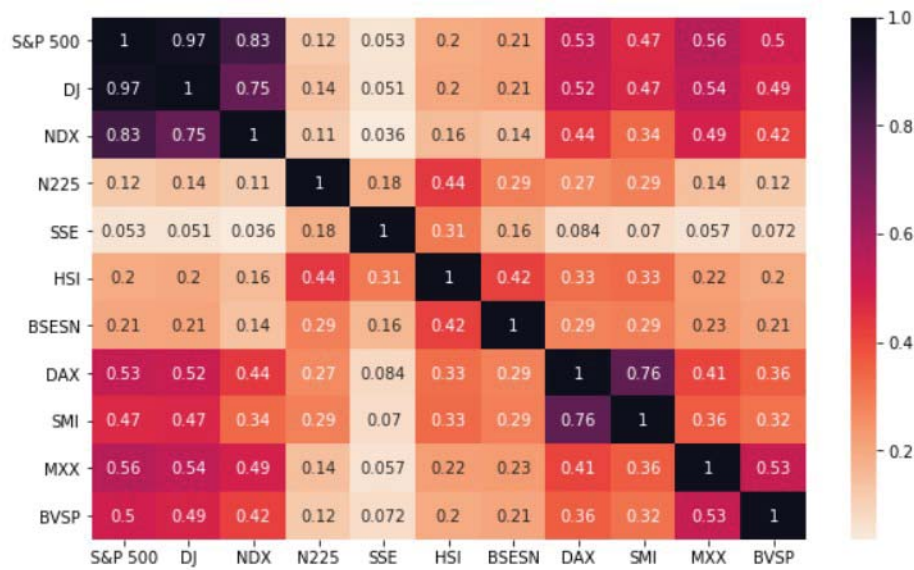


Figure 3.4: Correlation of daily returns of all eleven datasets

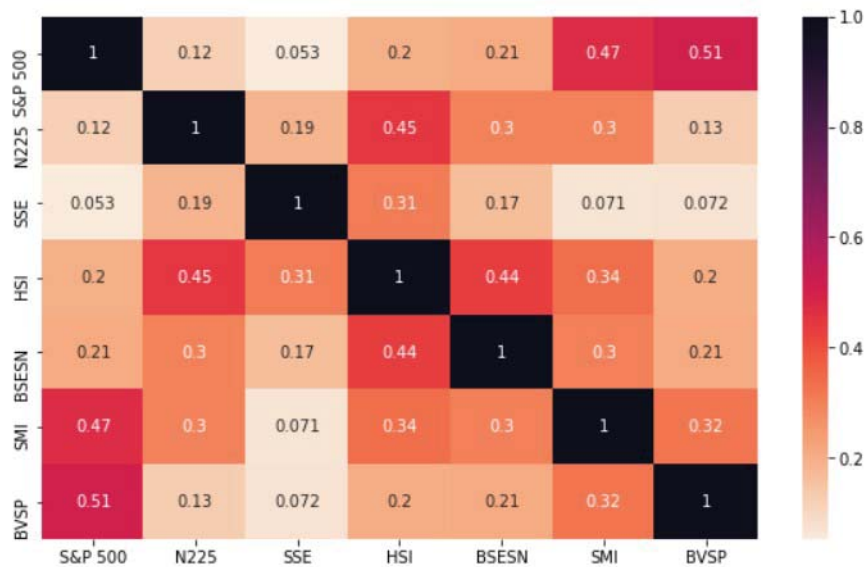


Figure 3.5: Least correlated datasets

The correlation matrices above indicates that S&P500, Nasdaq, DJI, and DAX,

SMI, and MXX, BVSP are strongly correlated. We have avoided the correlation of 0.5 for any two datasets so that it does not overfit during the training.

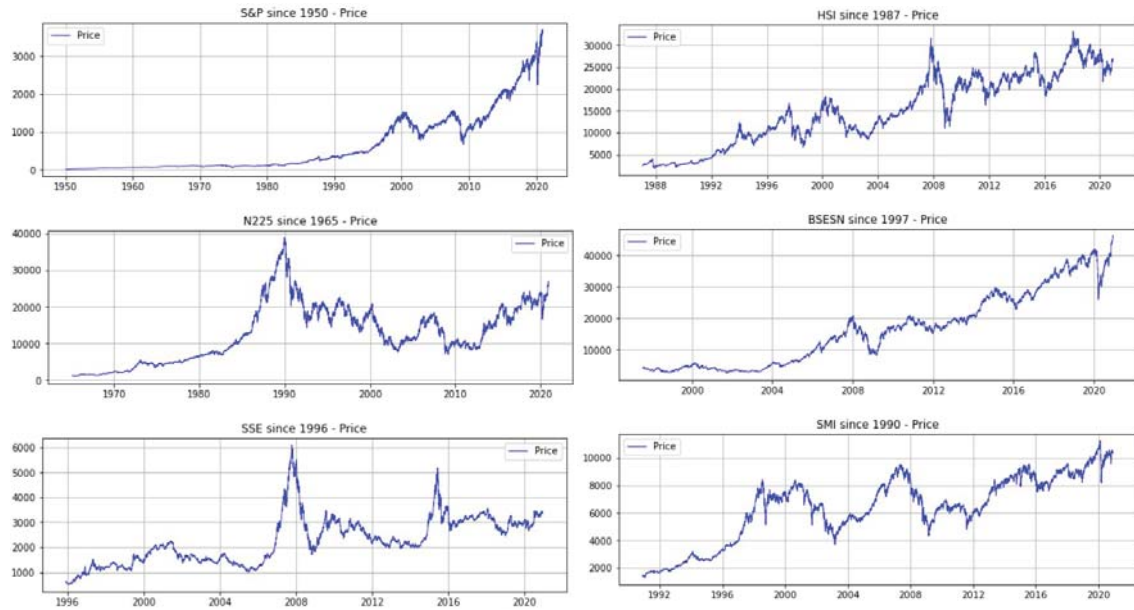


Figure 3.6: Time series plot distribution of daily price returns over the last years.

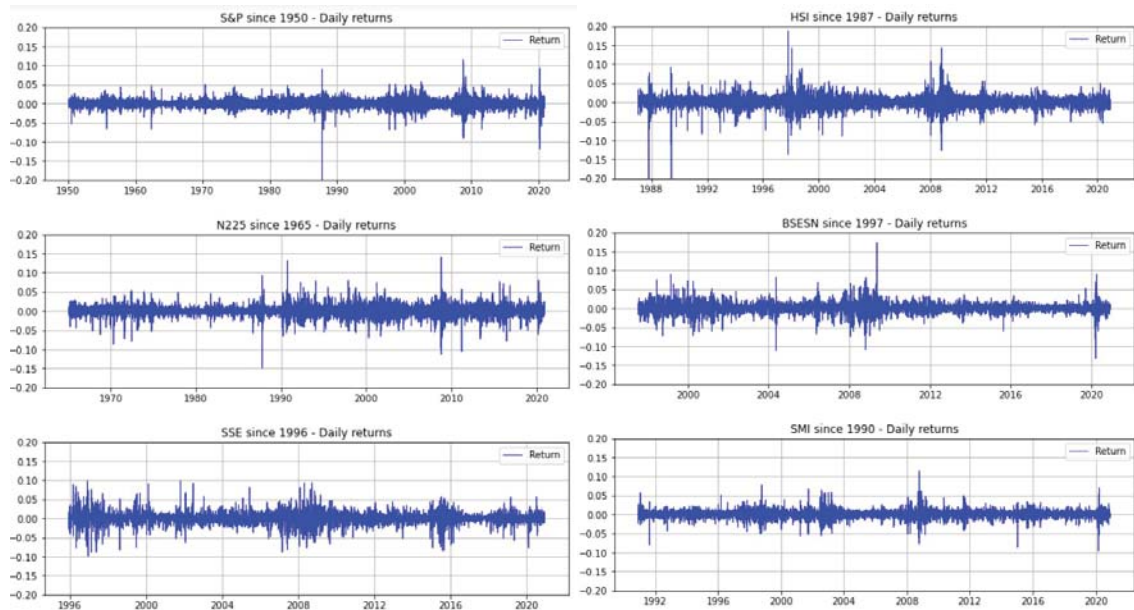


Figure 3.7: Fluctuation of daily returns indicating volatility of the markets

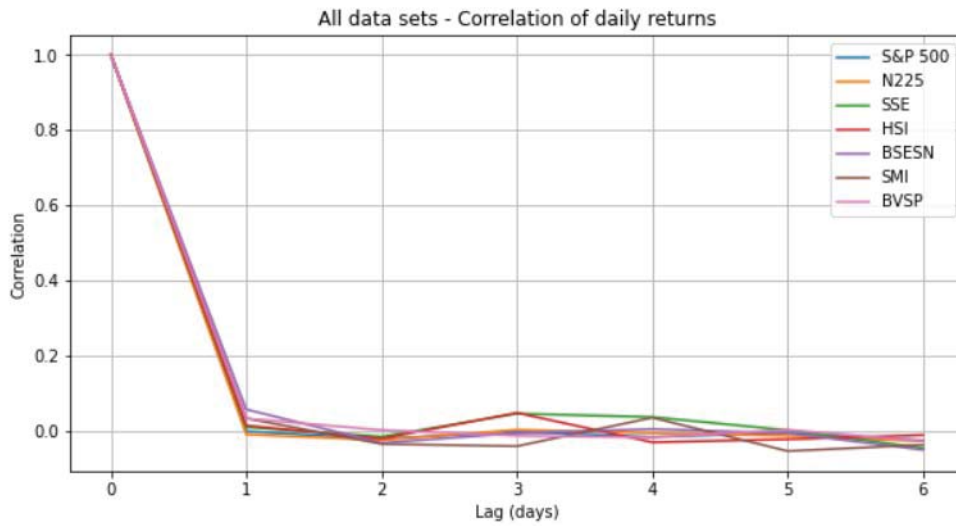


Figure 3.8: Auto-correlation plot, suggest that daily return can not be a strong indicator for the price change in the next following days.

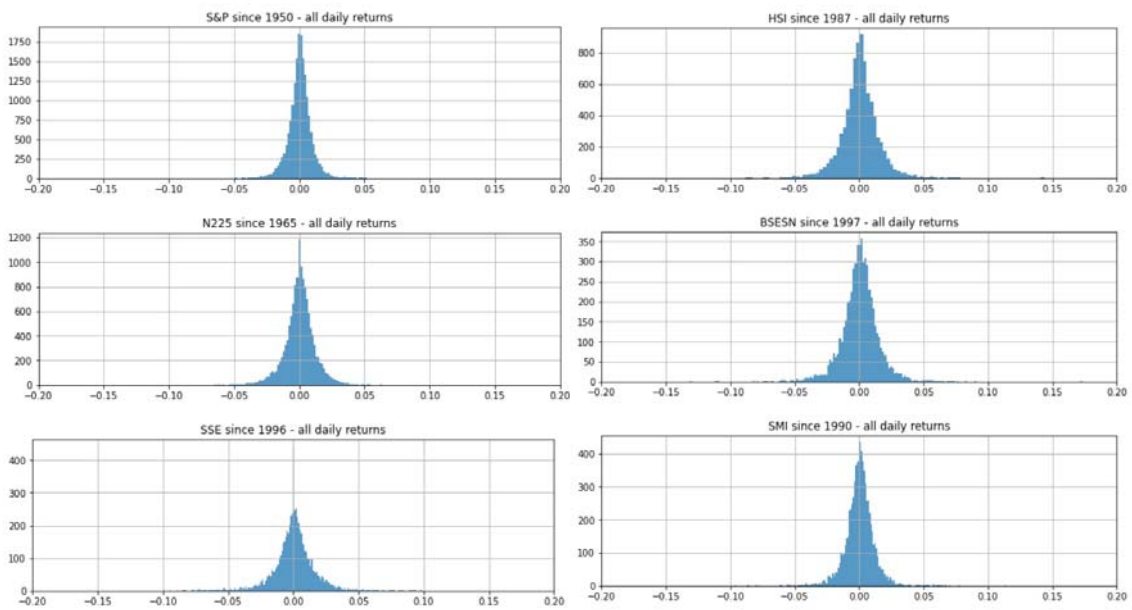


Figure 3.9: Distribution of daily returns. In SSE, there is fat tails indicating high volatility compared to others.

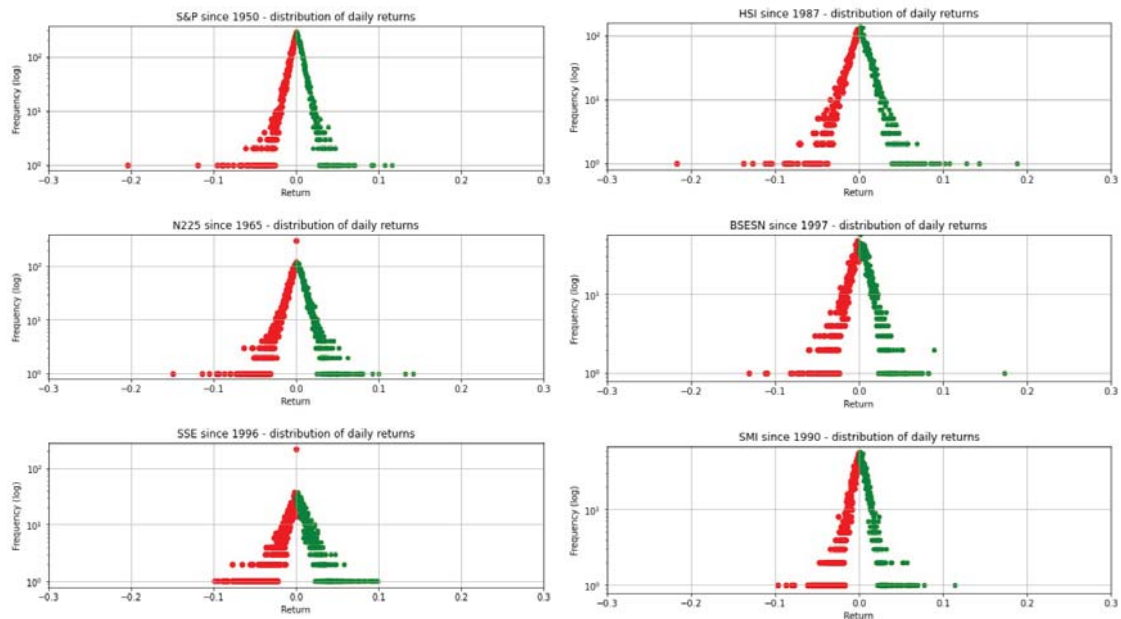


Figure 3.10: Frequency log-distribution of daily returns indicating extreme positive (green) and extreme negative (red) occurrence of return. Extreme negative returns less than -0.1 is more likely to contribute in a crash.

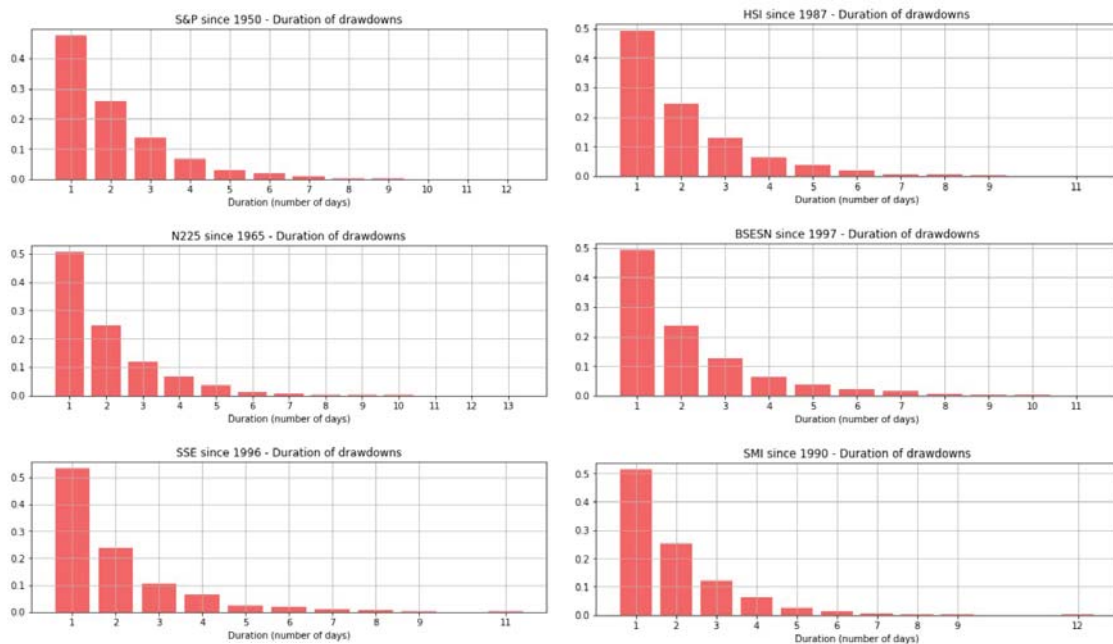


Figure 3.11: Duration of drawdowns.

On all six data-sets we have used, 50 per cent drawdowns hold out just a single day which determine that even if price is dropped in a day, the next day price again goes up. It supports the low auto-correlation mentioned above. The longest drawdowns take place about 10-12 business days.

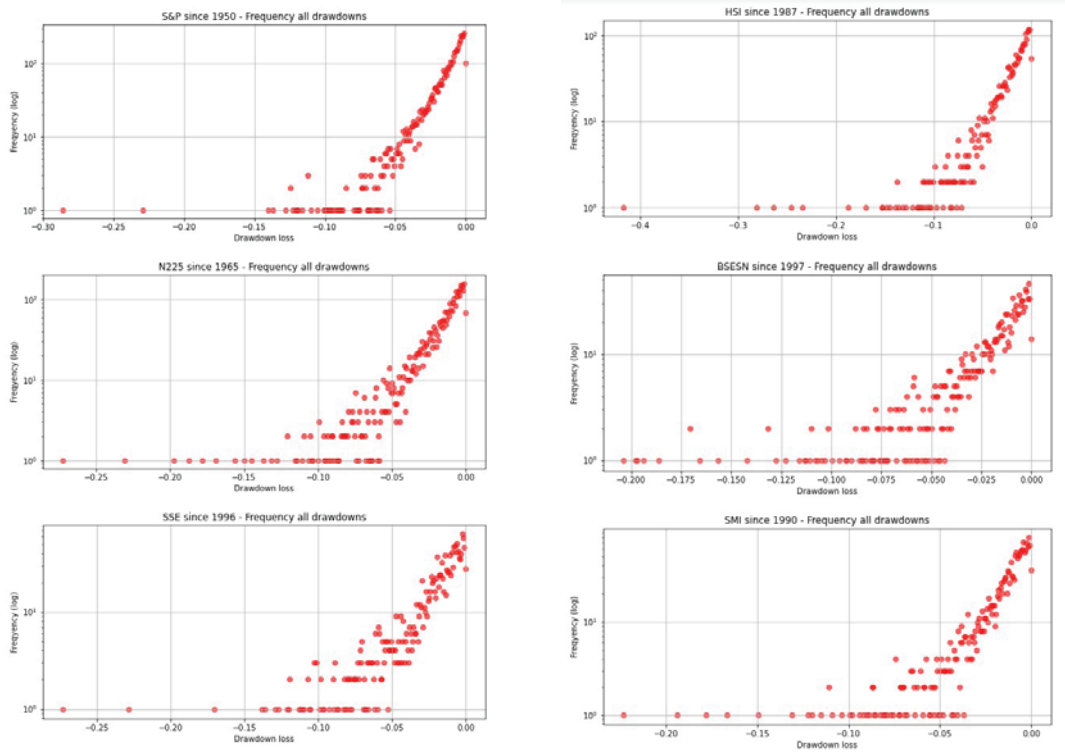


Figure 3.12: Frequency log-distribution of daily returns indicating extreme drawdowns (>-1.5) occurrence of return. Such significant drawdowns have only happened two times in the S&P for nearly 70 years. These extreme drawdowns are more likely associated with crashes.

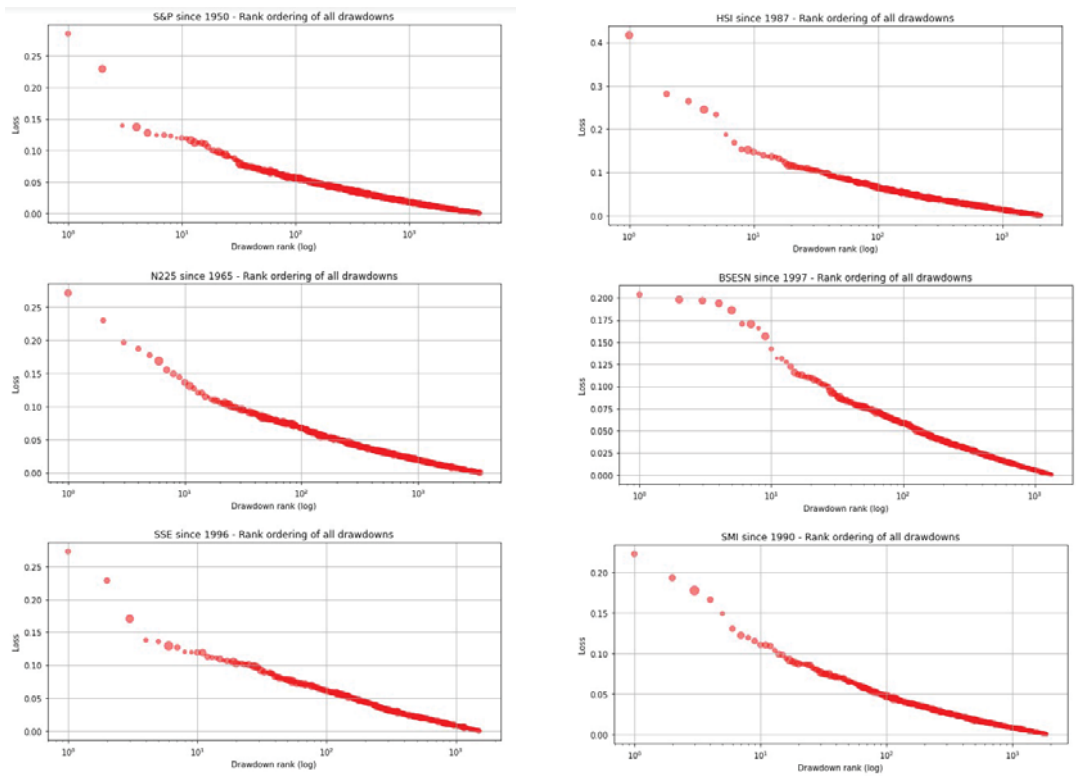


Figure 3.13: Drawdowns ranked from 1 to n.

The scale of each bubble corresponds to the length of each drawdown which indicates that the biggest drawdowns are not always the longest. These plots offer more visual proof of the presence of outliers as drawdowns which are greater than expected.

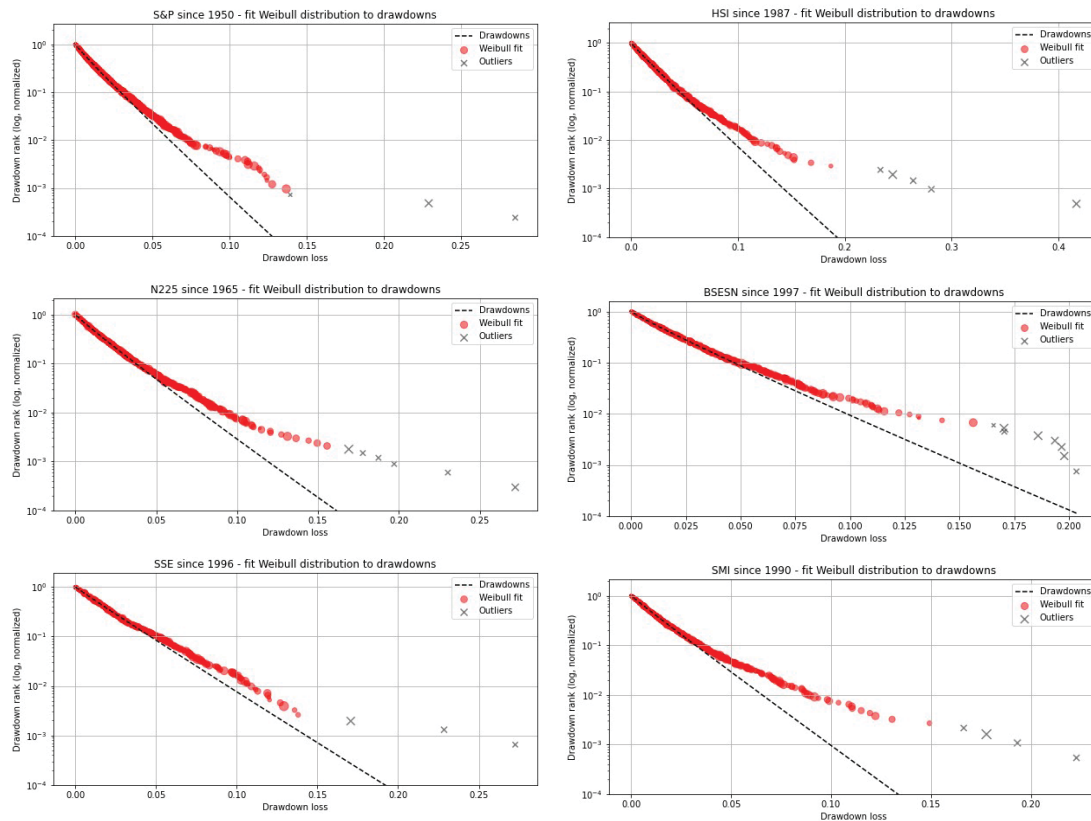
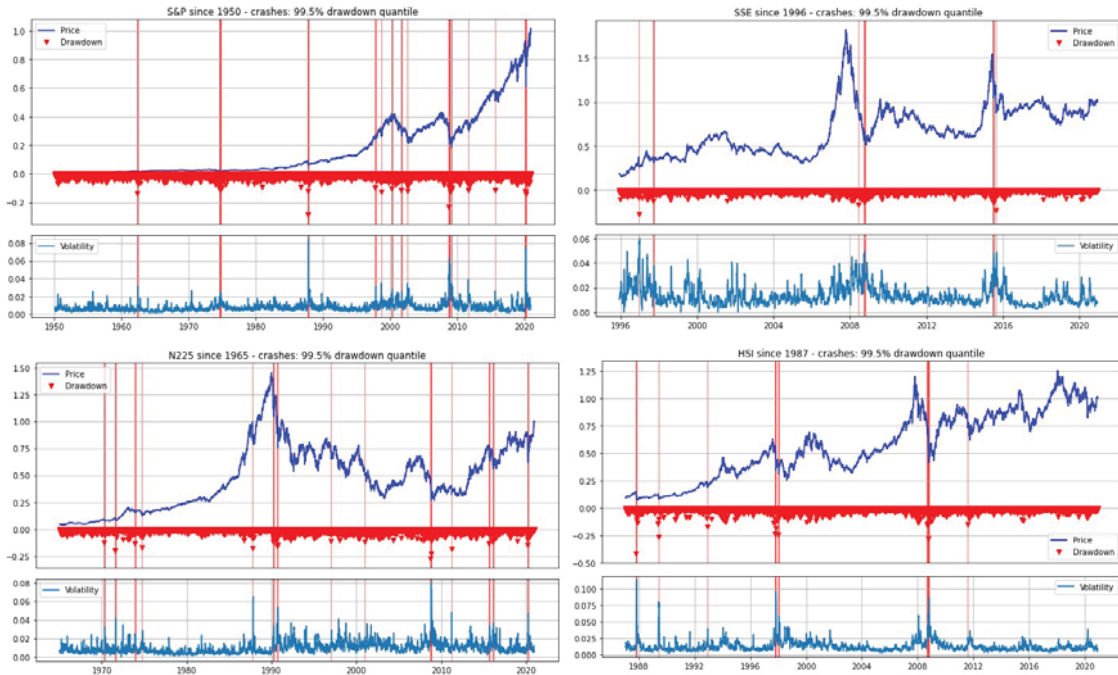


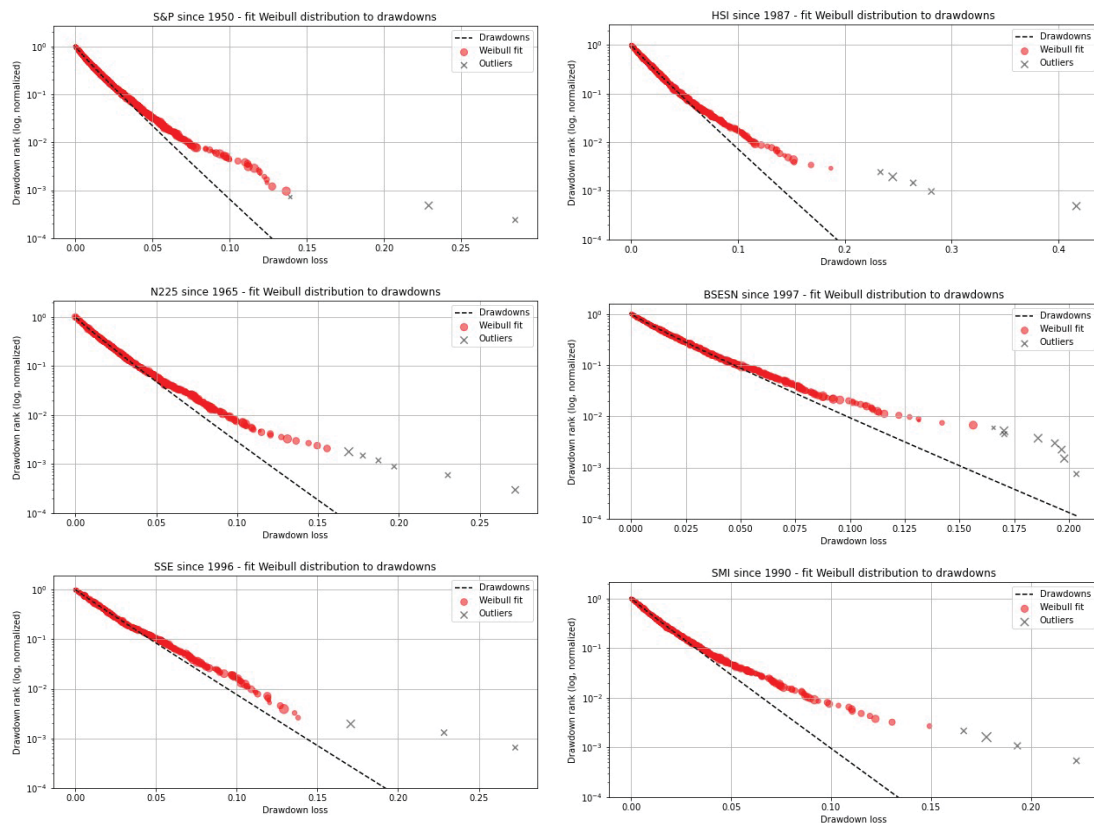
Figure 3.14: Weibull distribution by rank.

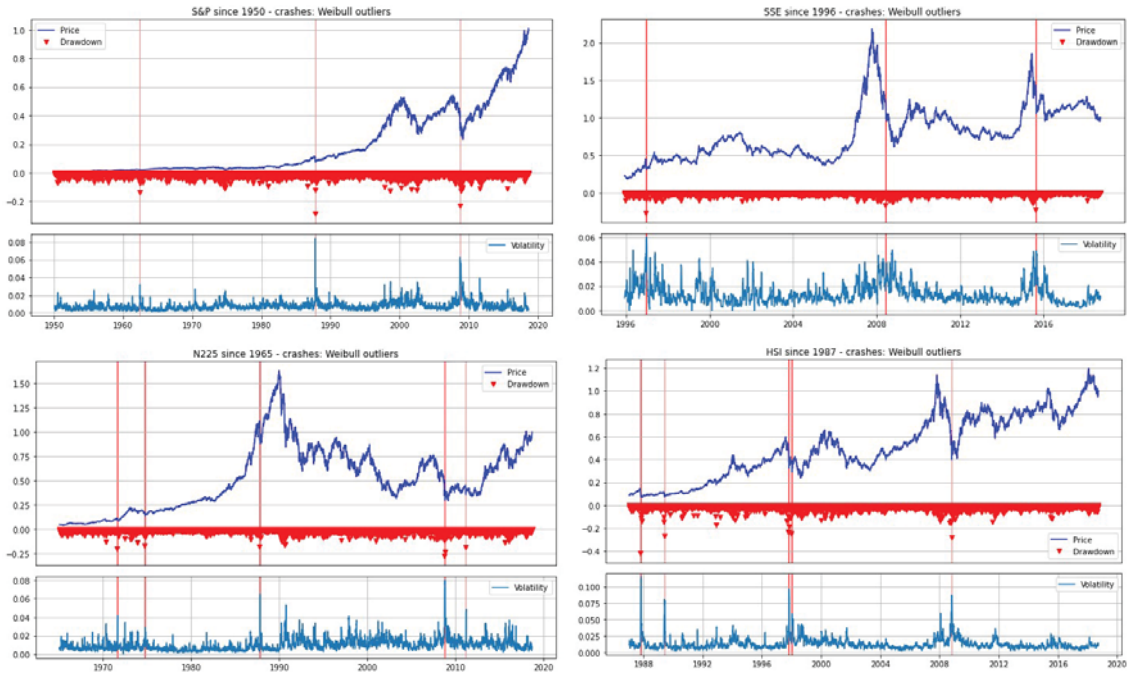
(Johansen, Sornette 2001) [4] have used $y \sim \exp(\frac{x}{\chi})^z$ to match and rank the drawdown. According to them, outliers are far away from the distribution which are nothing but market crash.

In the diagram the losses over 15% are identified as crash.



The Weibull fit plots below are the same as seen under 2. This time, though, with the "x"s defining outliers that cannot be clarified by the Weibull distribution. Since Johansen and Sornette do not imply a clear threshold divergence from the distribution that determines a crash, the detection of the above crashes was based on a visual understanding.





Identifying crashes on the basis of drawdown outliers results in a number of crashes that do not actually correlate to the average number of drawdowns in a dataset. For example, there are only three recorded crashes in the S&P over 68 years.

As there is no consensus about the precise concept of a financial collapse, both the method adopted by Jacobsson (99.5 percent quantile of drawdowns) and the method introduced by Johansen and Sornette (outliers found using the Weibull exponential model) can be used as an alternative to defining the crashes. we have used the quantile method as it has no manual interpretation of outliers required whereas we need to find outliers manually with the other theory.

Chapter 4

Model Description

4.1 Log Periodic Power Law

For the prediction of financial crises, we explain below how these theories can be translated into concrete form. In the log-periodic oscillations leading to a collision, the approach described here predicts the presence of a log-frequency shift over time. The following non-linear log-periodic formula can be used to fit comparatively long financial time-series (Sornette, D. 1997) [2]:

$$\log[p(t)] = A + B \frac{(t_c - t)^\beta}{\sqrt{1 + (\frac{t_c - t}{\delta_t})^{2\beta}}} \left[1 + C \cos \left[\omega \log(t_c - t) + \frac{\delta_w}{2\beta} \log \left(1 + \left(\frac{t_c - t}{\delta_t} \right)^{2\beta} \right) \right] \right] \quad (4.1)$$

A simple solution proposed by Johansen and Sornette is (Sornette, D. 1997) [1]

$$I(t) = A + B(t_c - t)^\alpha [1 + C \cos(\omega \log(t_c - t) - \phi)] \quad (4.2)$$

Here, $I(t)$ is the estimated log price at the bubble's termination date, α is the super exponential expansion, ϕ is the oscillations' time scale. A is the expected log price at the height when critical time is reached at the end of the bubble. B is the power law acceleration amplitude. C is the log-periodic oscillations amplitude. The critical time expected from the fit of each financial time series is t_c . The critical time t_c of the power law is not the time of the accident, but rather its most likely value, that is, the time at which the scaling ratio of the temporal oscillations hierarchy is the asymmetric distribution of the potential periods of the crash peaks.

A skewed random phenomena that happens with a probability that increases as time reaches critical time t_c . The incidence of the accident. Therefore, we assume that fits would give t_c values which are usually similar to but symmetrically later than the crash's real-time.

The fit is executed up to the t_{\max} time when the stock index hits its peak pre-crash maximum. The t_{\min} parameter is the time after the crash/rebound of the lowest point of the economy, disregarding the smaller plateau phases.

It is a delicate problem to suit data with sufficiently complicated formulae with a very large number of adjustable-parameters. We use the least-squares minimization approach to determine them analytically for the realistic application of the fit of a formula of some complex shape to a financial time sequence. This helps one to focus the linear parameters A , B and C and then construct an objective function by

plugging them into a focused objective function, which depends not only on t_c , β , ω , ϕ and, as in the case of a linear log-periodic formula, but also on two additional parameters: Δ_t and Δ_ω .

We want to remember that the non-linearity of the objective function will yield several local minima due to the noisy existence of financial data and the fact that we are doing a strongly non-linear 9 parametric fit. The fitting is then performed in a very intricate way. The way:

In the beginning, by decreasing the objective equation with respect to the three linear variables A, B, and C, the effective number of parameters is limited, thereby clearly determining A, B, and C as a function of the six nonlinear variables β , t_c , δ_t , ω , δ_ω , and ϕ . This procedure thus reduces the number of free variables in the fit from 9 to 6.

The safest approach we assume is to do a provisional search or first grid search using a number of appropriate values for t_c , δ_t , ω , δ_ω and fitting only β and ϕ . Due to the chance of the minimization algorithm being stuck in one of the several possible local minima. In fact, this will mean that δ_t does not have a value much greater or much smaller than the data time interval, as it calculates the characteristic time scale that governs the saturation and the crossover of the log frequency. In addition, since we do not suit noise, meaning variations on very small time-scales, it is not possible to consider very large values in our work for ω .

This is achieved in order to identify starting points for an optimizer, such as from all the local optima of the grid, the Levenberg-Marquardt. Because the exponent β must be between 0 and 1 for the price not just to accelerate but also to stay finite, all minimum satisfying $0 \leq \beta \leq 1$ will then be chosen after the search and taken as the starting values of fits with equation (1) to the data for all six non-linear free variables. Because the ϕ step is merely a time unit, however, which depends only on whether or not we have opted to take days, months or years into account, the fit is basically regulated by five non-linear parameters: β , t_c , δ_t , ω , δ_ω , and only the best resulting convergence points will be taken as the best fit. In other words, the solution with the nearest δ_t to the date time-interval will then be selected as the best match.

Since the transition time between two regimes is δ_t , this transition can be observed in a data set. Restrictions on parameter values are placed a priori, meaning that they are plausible. The stronger criteria $0.2 \leq \beta \leq 0.8$ was found to be useful in preventing the pathologies associated with the interval endpoints 0 and 1. The ratio of successive time-intervals between λ and successive time-intervals between local maxima is calculated by the angular log-periodic frequency ω by the following relationship:

$$\lambda = \exp \frac{2\pi}{\omega} \tag{4.3}$$

4.2 RNN LSTM (Statefull and Stateless)

The term ‘‘LSTM’’ stands for Long Short-Term Memory, meaning this neural network manages short-term memory.

Modes of LSTM:

For its optimal computation, there exist 2 main modes of LSTM; one is the stateful mode and the other is the stateless mode.

In an LSTM neural network, cell state is the cell memory of LSTM layer and hidden state is the state of the neurons in hidden layers of LSTM neural network. When, for better training, large datasets are split into batches for LSTM neural networks, in the stateless mode, LSTM updates parameters on first batch and then when the next batch comes, it initiates hidden states and cell states (meaning-memory), usually with zeros for the next batch. So, in the stateless LSTM mode, configuration starts from the beginning from batch to batch. After passing one batch, computations are going to be reset and initialized with zeros again before next batch starts and this way the process continues. On the other hand, in a stateful mode, the last output of the hidden state and the cell state from the first batch is used as the initial states or input for the next batch. So, it memorizes what it has learned in the first batch and then it takes it over to the next batch. It is these characteristics that differentiate between stateless and stateful modes of LSTM.

When to use which mode of LSTM?

When sequences in batches are related to each other, such as in prices of financial market indices, the stateful mode of LSTM could be the better choice in order to propagate the state of the neurons to the next batch for training instead of resetting it. As with large datasets for time series, there exists obvious dependency inside of the data, hence in between different time steps or different batches as well. So, time series are not independent in different batches and so, it is better to use stateful mode. Since, here we are working with stock market indices, stateful LSTM is often the recommended choice in order for the state to be propagated to the next batch.

For the other case, when one data sequence represents different complex structures within data (sentences, for instance), the recommended choice is the stateless mode of LSTM. With this stateless mode, the cell states for LSTM are reset at each sequence. Here, with one sequence in one batch and then another sequence in another batch, for two completely different sequences, they are completely unrelated to each other, so stateless mode is needed more instead of the stateful mode.

In our model, the stateful LSTM is performing slightly better than the stateless LSTM.

4.3 Decision Tree

Decision tree is a supervised machine learning algorithm that performs a series of sequential decisions at each node centered on a collection of features for data until a conclusion has been reached. Decision trees in supervised machine learning can handle both categorical and numerical data and so, both classification and regression models are used for the development of it. This algorithm can not only learn complex patterns from data but is also robust to noisy data. It non-linearly maps independent variables to dependent variable and is generally used for classifying non-linearly separable data. Unlike linear and logistic regression models, decision tree algorithm is more flexible for handling situations where the relationship between

features and outcome is nonlinear or, where features interact with each other.

Components of a decision tree:

Regression or classification models are built as tree-like constructs by the decision tree. The composition of a decision tree consists of a root node, internal or broken nodes, and nodes of a decision or leaf. Each tree node represents an instance attribute, and one of the possible values of that attribute is indicated by branches descending from that node. The root node corresponds to the top-level node leading to the best indicator in a tree. According to a set parameters, each test node then divides the data into further divisions from this node. In order to decide the best attribute to start with and divide the training data set, the discrete splitting function uses those parameters (Information gain, Gini index etc.). The attribute that better distinguishes ambiguity from target function information is known to be the most insightful. Different subsets of the data set are generated by slicing, with each instance belonging to a subset. The final subsets are called leaf nodes and broken nodes are called the intermediate subsets. A decision tree's key concept is to classify the features that provide the most information about the target feature and then divide the data several times according to such cut-off values in the features, making the target feature values as pure as possible for the resulting nodes. For most details, this search process proceeds until a tree with decision nodes and leaf nodes with leaf nodes representing a classification or a decision is the final outcome.

Information Gain

The measure through which the informativeness of features is given in order for the feature with the most information to be then used to split the data on is referred to as information gain. For finding the best feature in terms of information gain to use as root node, the data set is split along the values by using each descriptive feature right after entropy of the data set is calculated. By splitting data set along the feature values, remaining entropy is obtained and added proportionally for total entropy for split which is then subtracted from the calculated entropy of data set. This process is done in order to measure the original entropy reduced by this feature splitting, thus providing the information gain of a feature. The feature with maximum information gain is then chosen as root node to build the decision tree. The data set is divided by the branches of this root node and the same process is repeated on each branch (with entropy greater than zero) till leaf nodes (with entropy zero). Hence, information gain is based on entropy reduction from splitting data set based on a descriptive feature and a decision tree construction is nothing but finding attribute returning maximum information gain.

$$\text{InformationGain}(\text{feature}) = \text{Entropy}(\text{dataset}) - \text{Entropy}(\text{feature}) \quad (4.4)$$

Entropy

Entropy, which acts as a basis for the measurement of knowledge gain, is used to calculate a data set's impurity or randomness. To test the homogeneity of a sample, the ID3 learning-algorithm uses entropy. For the totally homogeneous sample, entropy is zero, but if the sample has homogeneous elements evenly separated, then entropy is one. As per the increase of impurity, thus purity decreases, entropy also increases. If a target function includes several element types, it is considered useful

to add up the entropy of each potential target value and then weigh it by the chance of having those values, given a random draw.

$$Entropy(x) = \sum (P(x = k)) * \log_2(P(x = k)) \quad (4.5)$$

Gini Index

The Classification and Regression Trees (CART) algorithm uses the Gini index for the construction of decision trees, which, unlike knowledge gain, prefers larger partitions and is simple to enforce. In order to compute this index, the sum of the squared percentages of each class is subtracted from one. The function with a lower Gini index is selected during splits. However, a wider range of results are obtained using Information Gain since entropy is used as its base calculation, whereas the Gini index simply caps at one.

$$GiniIndex = 1 - \sum (P(x = k))^2 \quad (4.6)$$

(Chien, TW. 2018) [16]

This is how a decision tree is created, which can be used in machine learning to forecast stock market patterns.

In order to accurately predict financial market index movements using decision trees in supervised machine learning using historical data and technical indicators of stock (after preventing look-ahead bias by lagging values of the technical indicator), a class column is added to data signifying daily returns based on adjusted close price of index. This class feature contains binary values for “up” and “down” denoting positive return and negative return, respectively. The data collection is first divided into two sections for training and checking the data in order to forecast the everyday shift in movements using these technical measures by adding a machine learning decision tree algorithm to the data set. The training data is then used to obtain information gain of features. After training, this decision tree algorithm is used on test data set to acquire probabilistic trend knowledge.

4.4 Linear Regression

Linear regression is a statistical learning method which determines the relationship between independent covariates and the dependent output variable.

How Linear Regression Works:

Simple linear regression model is used for showing or predicting relationship between two factors or variables, one dependent factor (the variable that is being predicted) and another is independent variable. A straight line is represented as the relationship between the two factors. When there are two or more variables, the model is called multiple regression.

Linear parametric regression models take the general form y to be equal to $f(x) + \epsilon$, where y is the predicted output variable, $f(x)$ is an unknown function and ϵ is the error term that is independent of the covariates.

$$y = f(x) + \epsilon \quad (4.7)$$

Now, different regression models use different forms of this function f . In Linear regression,

$$y = \beta_0 + \beta_1 x \quad (4.8)$$

This parametric form thus reduces the problem of finding a relationship between the covariates or independent variable X and the response variable y to only determining 2 coefficients which are β_0 and β_1 . Linear regression makes an estimate of these coefficients where such estimates are represented with a “hat” on the variable:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \quad (4.9)$$

Using this method, the values of these coefficients are determined based on given data.

It is to be noted that, the epsilon term is not a part of this particular equation because epsilon is independent of the covariate X and so, it cannot be determined by regression analysis. So, even if we make a perfect linear regression model, the model would still have some error epsilon which is also known as “irreducible error”. Now, in order to estimate these coefficients, we want to minimize the difference between the actual output value y and the predicted output value \hat{y} by the model for every sample X . This difference is called the “residual error” –

$$e = y - \hat{y}$$

So,

$$e = \beta_0 + \beta_1 \cdot X + \epsilon - (\hat{\beta}_0 + \hat{\beta}_1 \cdot X) \quad (4.10)$$

So, by definition, epsilon is part of the residual error. This regression method is used when prediction of an outcome variable is needed based on the value of covariates.

4.5 Logistic Regression

Logistic Regression model is a predictive algorithm that uses independent factors to predict the dependent factor. Logistic regression in machine learning aims is to find probabilities for some actions rather than adjustments in simple regression case. The basic regression methods and the logistic regression methods both seek to find the best fitting line for the given data - or curve in the case of logistic regression.

How Logistic Regression Works:

When the dependent variable is binary, the logistic regression model is the best regression analysis model to use. In order to interpret data and illustrate the relationship between one dependent binary variable and one or more variables that may be nominal, ordinal, interval or independent ratio level, logistic regression is performed. (Gasso, 2019) [19].

The logistic regression model is derived by –

$$Y = b_0 + b_1 * X$$

$$P = \frac{1}{1 + e^{-Y}}$$

$$\ln\left(\frac{P}{1-P}\right) = b_0 + b_1 * X$$

Here, P - probability of a 1 (proportion of 1s, mean of Y), e - e base of natural log which value is 2.718) and a and b are the parameters. The relationship between P and X is non-linear.

Loss Function:

Loss function calculates the fit between real data and data from a mathematical model. The parameters are chosen in a way so that the model minimizes the inaccuracy or maximizes the accuracy to the data. With least squares which is denoted as *SS_res* which is called the sum of squares residual. In logistic regression a mathematical solution exists that will minimize the sum of squares to maximize the accuracy of the model, that is,

$$b = (X'X)^{-1}X'y \quad (4.11)$$

When To Use Logistic Regression:

Logistic regression should be used when there are model multiple independent variables, continuous and categorical variables or polynomial terms to model curvature. To assess interaction terms in determining if the effect of one independent variable depends on the value of another variable, logistic regression model is used.

4.6 Back Propagation

The back propagation is a multi-layered, feed-forward artificial neural network, which is the most commonly used by far. Let us explain the whole process of the algorithm by a simple 1-1-1 network at figure [11] with one input layer (L-2), one hidden layer (L-1) and one output layer (L). Each neuron at hidden layer and output layer has three basic parts shown in figure [3]. We denoted weighted sum or input function by 'z' and output function by 'y'. We have used the sigmoid function as activation here. The reason behind using sigmoid function will be discussed later.

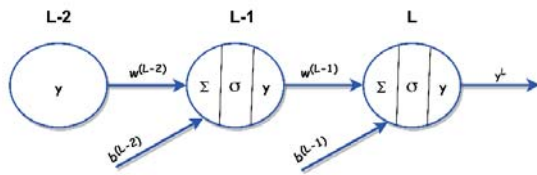


Figure 4.1: Simple neural network

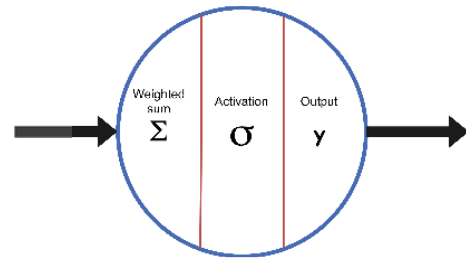


Figure 4.2: Basic structure of a neuron

Forward pass:

Weighted sum: To get the weighted sum, $z^{(L-1)}(n)$ of the neuron in (L-1) layer, we have multiplied output, $y^{(L-2)}(n)$ of the (L-2) layer and corresponding weight, $\omega^{(L-2)}(n)$ connected from (L-2) layer to (L-1) layer. After that, we have added the bias term, $b^{(L-2)}(n)$ of the layer (L-2). Here (n) is the iteration number.

$$z^{(L-1)}(n) = \omega^{(L-2)}(n) * y^{(L-2)}(n) + b^{(L-2)}(n) \quad (4.12)$$

Activation Function: In multilayer perceptron differentiability is the requirement that an activation function must satisfy in order to propagate easily [14]. The most commonly used differentiable non-linear activation function is sigmoidal function [14].

$$\sigma(z^{(L-1)}(n)) = \frac{1}{1 + e^{(-z^{(L-1)}(n))}} \quad (4.13)$$

Output Function: In equation 1.2, the weighted sum calculated on the eqn. 1.1 passed through a non linear activation function in order to get the output function which will be the input for the next layer.

$$y^{(L-1)}(n) = \sigma(z^{(L-1)}(n)) \quad (4.14)$$

Now the similar method will be applied to calculate weighted sum, $z^{(L)}(n)$ and output function, $y^{(L)}(n)$ of the final layer (L).

$$z^L(n) = \omega^{(L-1)}(n) * y^{(L-1)}(n) + b^{(L-1)}(n) \quad (4.15)$$

$$\sigma(z^{(L)}(n)) = \frac{1}{1 + e^{(-z^{(L)}(n))}} \quad (4.16)$$

$$y^L(n) = \sigma(z^L(n)) \quad (4.17)$$

Now we have the final or the predicted output, $y^{(L)}(n)$ and we need to compare it with the desire output, d^L to calculate the cost function also known as loss function $C(d^L, y^{(L)}(n))$. It may be noted, we did not put (n) with the desired output. The reason behind is, in supervised learning, the desire output always remains the same, it does not change in every iteration like the predicted output, $y^{(L)}(n)$.

The cost function, C measures how well the model is. The two most common cost functions are Mean-Square-Error (MSE) and Cross-entropy loss (log loss). We have used MSE for our back-propagation algorithm. The general formula of Mean Square Error (MSE) is:

$$C(d^L, y^L) = \frac{1}{N} \sum_{(i=1)}^N (d^L - y^L)^2 \quad (4.18)$$

Here, N=number of sample or number of neurons at the final layer

In our simple 1-1-1 network the cost function is:

$$C(d^L, y^L(n)) = \frac{1}{2} (d^L - y^L)^2 \quad (4.19)$$

Back Propagation:

Now the weights will be updated in order to get closer to the desired output, d^L based on the cost value. The chain rule will be used here to compute the gradient descent of the cost function with respect to the weights. In machine learning, Gradient descent is used for optimization. The equation [1.9] calculates the gradient descent of the cost w.r.t the weights of the final layer(L). Similarly, equation [2.0] calculates the gradient descent of the bias of the final layer (L).

$$\frac{\delta C(n)}{\delta w^{(L-1)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y^L(n)} * \delta y^L(n) \right)}{\delta z^L(n)} * \delta z^L(n) \quad (4.20)$$

$$\frac{\delta C(n)}{\delta b^{(L-1)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y^{(L)}(n)}\right) * \delta y^{(L)}(n)}{\delta z^{(L)}(n)} * \delta z^{(L)}(n) \quad (4.21)$$

The new updated weight and bias of the layer (L-1) to layer (L) is:

$$\omega'^{(L-1)}(n) = w^{(L-1)}(n) - \eta \left(\frac{\delta C(n)}{\delta w^{(L-1)}(n)} \right) \quad (4.22)$$

$$b'^{(L-1)}(n) = w^{(L-1)}(n) - \eta \left(\frac{\delta C(n)}{\delta b^{(L-1)}(n)} \right) \quad (4.23)$$

There is a problem with the above two updated formulas [2.1] and [2.2]. We have introduced a new term, η which is called learning rate. Learning rate decides how faster or slower the network will be trained. The learning rate normally lies between 0.1 to 0.9. The problem here is, how to decide the value of learning rate? If we choose η small, the changes in the weights will be smaller. On the other hand if we choose η large, the network may become unstable (oscillatory) [1]. Although it depends largely on the application, Haykin (2010)[10], suggests a simple method of increasing the rate of learning by avoiding the danger of instability is to modify the delta rule by including a momentum constant, α (a positive number).

The delta rule is,

$$\Delta w(n) = -\eta \frac{\delta C(n)}{\delta w(n)} \quad (4.24)$$

Here $\Delta w(n)$ is the change of weight at the present iteration. The use of minus (-) is to account for gradient descent in weight space.

The new delta rule is,

$$\Delta w(n) = \alpha \Delta w(n-1) + \eta \frac{\delta C(n)}{\delta w(n)} \quad (4.25)$$

Here, $\Delta w(n-1)$ is the change of weight at the previous iteration.

After applying the generalized delta rule in our equation [2.1] and [2.2]

$$\omega'^{(L-1)}(n) = w^{(L-1)}(n) + \Delta w^{(L-1)}(n) \quad (4.26)$$

$$b'^{(L-1)}(n) = b^{(L-1)}(n) + \Delta b^{(L-1)}(n) \quad (4.27)$$

Now let us apply the same method for the layer (L-2),

Gradient descent of cost w.r.t weight,

$$\frac{\delta C(n)}{\delta w^{(L-2)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y^{(L-1)}(n)}\right) * \delta y^{(L-1)}(n)}{\delta z^{(L-1)}(n)} * \delta z^{(L-1)}(n) \quad (4.28)$$

$$\frac{\delta C(n)}{\delta b^{(L-2)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y^{(L-1)}(n)}\right) * \delta y^{(L-1)}(n)}{\delta z^{(L-1)}(n)} * \delta z^{(L-1)}(n) \quad (4.29)$$

The updated weight and bias,

$$\omega'^{(L-2)}(n) = w^{(L-2)}(n) + \Delta w^{(L-2)}(n) \quad (4.30)$$

$$b'^{(L-2)}(n) = b^{(L-2)}(n) + \Delta b^{(L-2)}(n) \quad (4.31)$$

After adjusting all the weights and biases of the network, the forward and backward pass will repeat in the same way until the model gets closer to the desired output. So far, we know how the back propagation works for a 1-1-1 network. Now we have demonstrated the whole process again shortly for an arbitrarily large network for better understanding. We have also generalized all the required formulas so that they work for all kinds of topology.

As an example, for our demonstration, let us consider a network of 3 layers where each layer consists of m number of nodes.

Forward Pass:

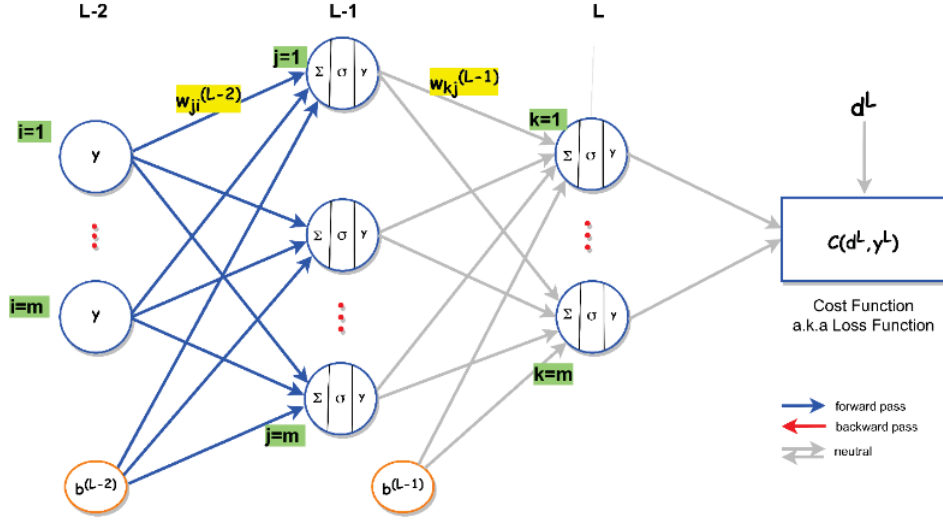


Figure 4.3: calculation of weighted sum and activation of layer (L-1)

In the network at 4.3, the weighted sum and the output function of layer (L-1) are,

$$z_j^{(L-1)}(n) = \sum_{(i=1)}^m \omega_{ji}^{(L-2)}(n) * y_j^{(L-2)}(n) + b_j^{(L-2)}(n) \quad (4.32)$$

$$y_j^{(L-1)}(n) = \sigma z_j^{(L-1)}(n) \quad (4.33)$$

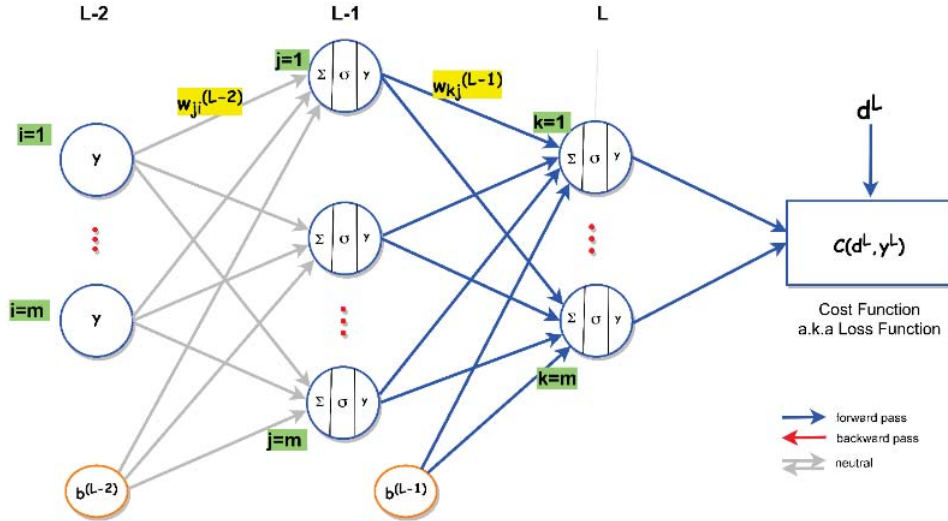


Figure 4.4: calculating weighted sum and activation of final layer (L). After that calculating the cost function

the weighted sum and the output function of layer (L) are,

$$z_k^L(n) = \sum_{(j=1)}^m \omega_{kj}^{(L-1)}(n) * y_j^{(L-1)}(n) + b_k^{(L-1)}(n) \quad (4.34)$$

$$y_k^L(n) = \sigma z_k^L(n) \quad (4.35)$$

Now, the formula for calculating the cost function is,

$$C(d^L, y_k^L) = \frac{1}{N} \sum_{(k=1)}^N (d^L - y_k^L)^2 \quad (4.36)$$

Backward pass:

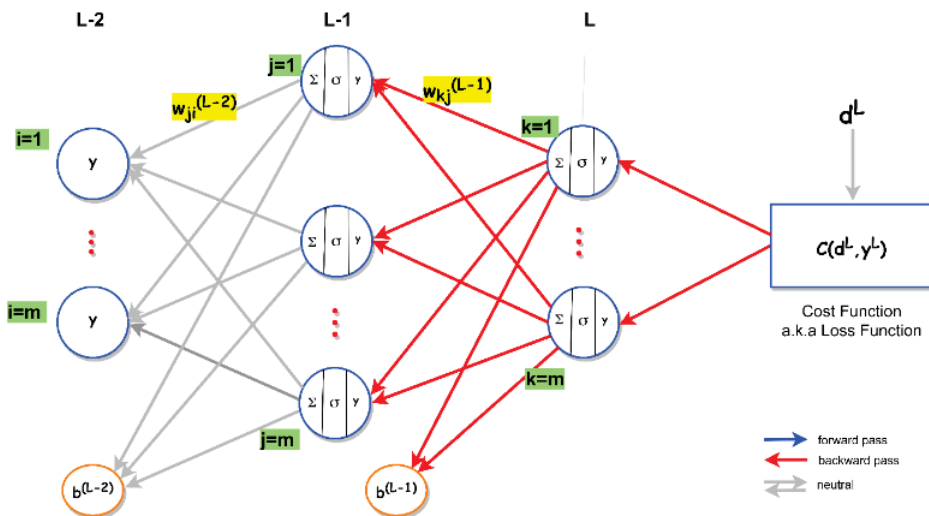


Figure 4.5: error signal from the layer (L) to the layer (L-1)

calculate the gradient descent of the cost w.r.t the weights and bias of the final layer(L),

$$\frac{\delta C(n)}{\delta w_{kj}^{(L-1)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y_k^L(n)}\right) * \delta y_k^L(n)}{\delta z_k^L(n)} * \delta z_k^L(N) \quad (4.37)$$

$$\frac{\delta C(n)}{\delta b_{kj}^{(L-1)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y_k^L(n)}\right) * \delta y_k^L(n)}{\delta b_{kj}^{(L-1)}(n)} * \delta z_k^L(N) \quad (4.38)$$

Change of weight and bias according to the generalized delta rule,

$$\Delta w_{kj}^{(L-1)}(n) = \alpha \Delta w_{kj}^{(L-1)}(n) + \eta \left(\frac{\delta C(n)}{\delta w_{kj}^{(L-1)}(n)} \right) \quad (4.39)$$

$$\Delta b_k^{(L-1)}(n) = \alpha \Delta b_k^{(L-1)}(n) + \eta \left(\frac{\delta C(n)}{\delta b_k^{(L-1)}(n)} \right) \quad (4.40)$$

New updated weight and bias of the layer (L-1),

$$\omega_{kj}^{(L-1)}(n) = w_{kj}^{(L-1)}(n) - \eta \left(\frac{C(n)}{\delta w_{kj}^{(L-1)}(n)} \right) \quad (4.41)$$

$$b_k^{(L-1)}(n) = b_k^{(L-1)}(n) + \Delta b_k^{(L-1)}(n) \quad (4.42)$$

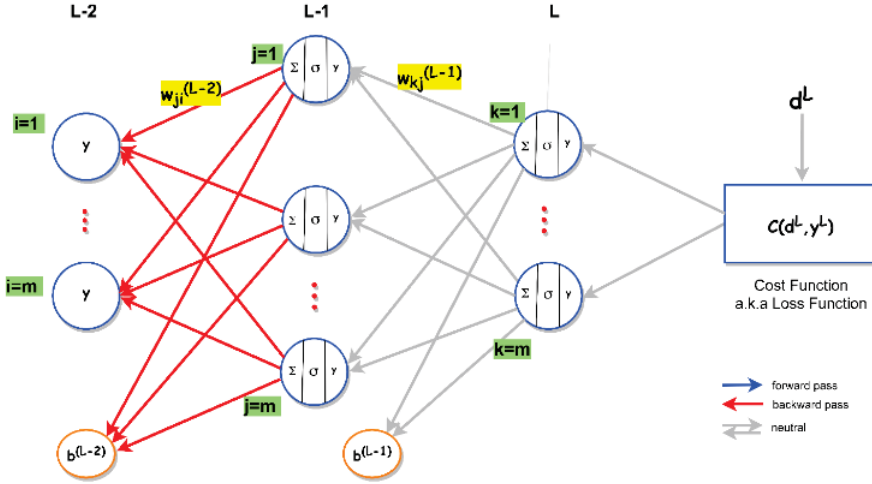


Figure 4.6: error signal from the layer (L-1) to the layer (L-2)

calculating the gradient descent of the cost w.r.t the weights and bias of the final layer(L-1),

$$\frac{\delta C(n)}{\delta w_{ji}^{(L-2)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y_j^{(L-1)}(n)}\right) * \delta y_j^{(L-1)}(n)}{\delta z_j^{(L-1)}(n)} * \delta z_j^{(L-1)}(N) \quad (4.43)$$

$$\frac{\delta C(n)}{\delta b_j^{(L-2)}(n)} = \frac{\left(\frac{\delta C(n)}{\delta y_k^{(L-1)}(n)}\right) * \delta y_k^{(L-1)}(n)}{\delta z_k^{(L-1)}(n)} * \delta z_k^{(L-1)}(N) \quad (4.44)$$

Change of weight and bias according to the generalized delta rule,

$$\Delta w_{ji}^{(L-2)}(n) = \alpha \Delta w_{ji}^{(L-2)}(n-1) + \eta \left(\frac{\delta C(n)}{\delta w_{ji}^{(L-2)}(n)} \right) \quad (4.45)$$

$$\Delta b_j^{(L-2)}(n) = \alpha \Delta b_j^{(L-2)}(n-1) + \eta \left(\frac{\delta C(n)}{\delta b_j^{(L-2)}(n)} \right) \quad (4.46)$$

New updated weight and bias of the layer (L-2),

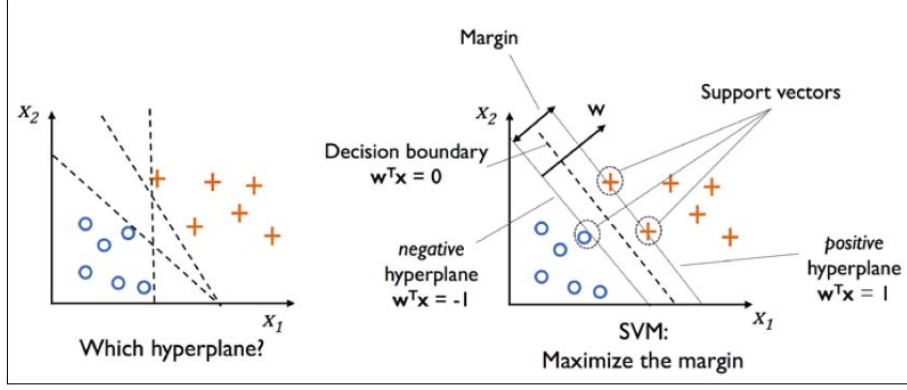
$$\omega_{ji}'^{(L-2)}(n) = w_{ji}^{(L-2)}(n) - \eta \left(\frac{C(n)}{\delta w_{ji}^{(L-2)}(n)} \right) \quad (4.47)$$

$$b_j'^{(L-2)}(n) = b_j^{(L-2)}(n) + \Delta b_j^{(L-2)}(n) \quad (4.48)$$

We have updated all the weights and bias of the network successfully for one iteration only. The process will continue until the model reaches the sufficient small amount of error.

4.7 Support Vector machine

”A support vector machine (SVM) is a supervised machine learning algorithm that does data analyzing which is used for classification the most as well as for regression analysis” (Patil,S. 2016) [12]. It uses the kernel technique for data transformation. Then, according to the transformed data, An optimal boundary is prepared by SVM between the probable outputs. In order to categorize data points, SVM maps data to a higher dimensional feature space because it is necessary even when the data set is linearly not separable. ”In order to transform the data, a category separator is to be found to be drawn as a hyper-plane” (Henrique,S. 2018) [17]. Initially each data item is plotted as a point in n-dimensional space. n denotes as the number of features present. Then a hyper-plane is to be found to classify the data points such that it differentiates the classes with as much accuracy as possible. The dimension of the hyper-plane is n-1. SVM algorithm finds a plane that separates the positive and negative points in the most accurate way. The distance between the planes is called margin which should be maximum.



Here, we can see there are two different classes and the circle selected points are called support vectors. "Support vectors are data points closer to the hyper-plane. Support vectors can influence the position and orientation of the hyper-plane" (Patil,S. 2016) [12]. Support vectors can maximize the margin (which is the distance between the hyper-planes) of the classifier. There are multiple hyper-planes, one positive hyper-plane $W^T * X = 1$ and one negative hyper-plane $W^T * X = -1$. The hyper-plane between the positive and negative hyper-plane is the decision boundary where $W^T * X = 0$. In order to reduce generalization error the decision boundary with larger margins should be chosen because smaller margins may cause over-fitting (outlier data).

Maximization of Margin:

$$W^T * X_{pos} + b = 1 \quad (4.49)$$

$$W^T * X_{neg} + b = -1 \quad (4.50)$$

Here are two equations of positive and negative hyper-planes where b is a new addition. b is the y-intercept. Now, subtracting equation (2) from equation (1),

$$W^T * (X_{pos} - X_{neg}) = 2 \quad (4.51)$$

Normalizing by the length of W ,

$$\|W\| = \sqrt{\sum W_j^2} \quad (4.52)$$

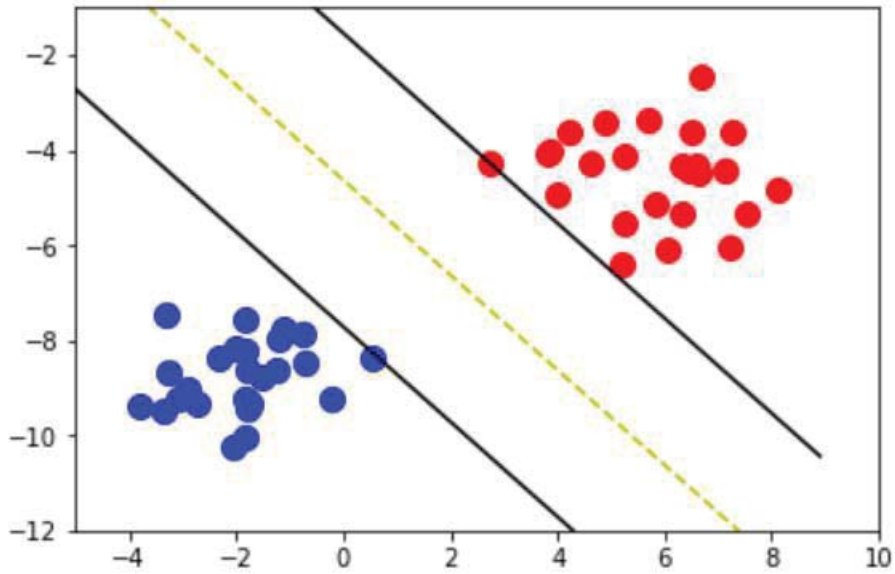
The final equation is,

$$\frac{W^T (X_{pos} - X_{neg})}{\|W\|} = \frac{2}{\|W\|} \quad (4.53)$$

Here, the left side of the equation is the margin.

Optimization Function:

$Argmax(2/\|W\|)$ for all i such that $Y(W^T * X_i + b) \geq 1$. This indicates all the data points of being separable linearly.



This is the hard-margin SVM, hardly used in real life problems.

Soft-Margin SVM:

The soft-margin SVM equation is:

$$\text{Argmin}\left(\frac{\|W\|}{2}\right) + C * \frac{1}{n} * \sum_{(i=1)}^n \zeta_i \quad (4.54)$$

Here, $\|W\|/2$ is used instead of $2/\|W\|$. As the value is inverted, argmin is used instead of argmax. n resembles the number of datapoints, C denotes as a hyper-parameter and ζ (Zeta) is the distance between points that are not classified correctly.

Understanding ζ (Zeta):



For point x_1 , distance between x_1 and plane Π is 0.5.

$Y(W^T * X + b) = -0.5$ (Minus indicates it is towards negative plane)

$$Y(W^T * X + b) = 1 - 1.5$$

$$Y(W^T * X + b) = 1 - \zeta(\text{Zeta})$$

Understanding C:

As hyperparameter C increases, overfit increases. Similarly, as C decreases, underfit increases. "Overfitting is a modeling error that happens when a function matches too closely to a small number of data points. On the other hand, underfitting is a modeling error that occurs when unable to capture the trend of data which means the model or algorithm is not well fitted to the data" (Kecman,V. 2005) [6].

Why Use +1 and -1:

$$\Pi^+ : W^T * X_{pos} + b = k$$

$$\Pi^- : W^T * X_{neg} + b = -k$$

Here, $k > 0$. It is not mandatory to use +1 and -1 as the value of k. But distinct values for the planes (i.e.: +k1 and -k2) cannot be chosen because The goal is for the planes to be equally remote. So the updated margin is, $2 * k / \Pi$. So the new margin is, $2 * k / \|W\|$. For $k = 5$, the margin is, $10 / \|W\|$. Since k is a constant, it will not affect the optimization of the problem. This is why +1 and -1 values are used for the simplification of the calculation.

Loss Function:

In Support Vector Machines, the loss function used is called hinge loss.

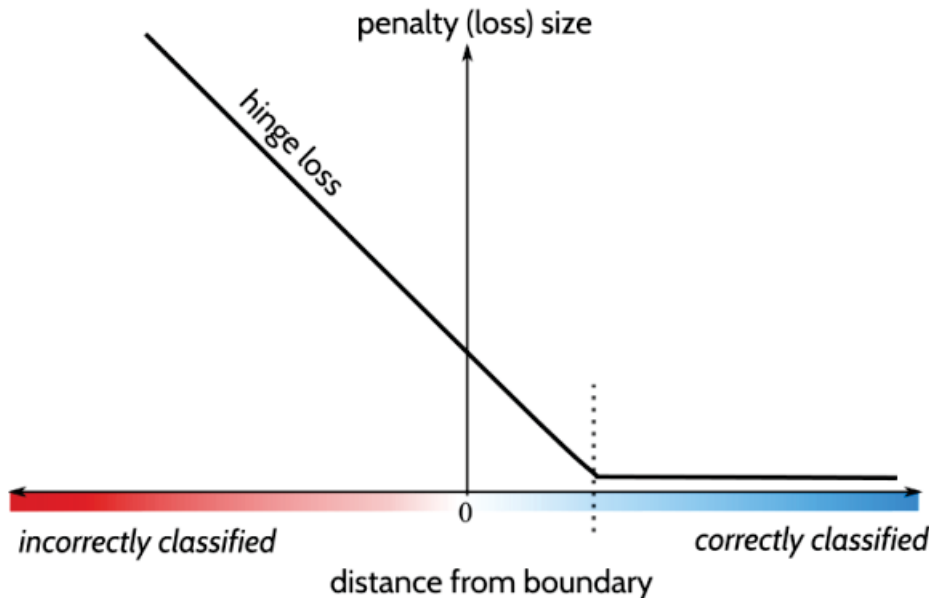


Figure 4.7: calculating weighted sum and activation of final layer (L). After that calculating the cost function

Hinge loss is a function of non-zero value till a certain point (let, the certain point is z). "A hinge failure is used by SVM since it highlights the boundary points conceptually" (Noble,D. 2006) [7].Because of the "hinge" (the max) in the function,

something further than the closest points adds little to the loss. It then reduces to choosing a boundary that yields the greatest margin (distance to closest point). The equation for soft-margin SVM,

$$\text{Argmin}\left(\frac{\|W\|}{2}\right) + C * \frac{1}{n} * \sum_{(i=1)}^n \zeta_i \quad (4.55)$$

Here, the second term $C * \frac{1}{n} * \sum_{(i=1)}^n \zeta_i$ is called a loss term.

$$Y(W^T * X + b) = Z.$$

If $Z \geq 1$ the classification is correct, but if $Z < 1$, the argument is incorrectly classified.

Let, there are two points, x1 (positive) and x2 (negative). x1 lies in the positive plane and x2 in the negative. For point x1, $W^T * X + b$ is a positive value and $Y(\text{classlabel}) = +1$.

$$\text{So, } Y(W^T * X + b) = +1 * (+\text{vevalue}) = +\text{vevalue}$$

$$\text{Similarly, for x2, } Y(W^T * X + b) = -1 * (-\text{vevalue}) = +\text{vevalue}$$

$$\text{Now, } Y(W^T * X + b) = +1 * (-\text{vevalue}) = -\text{vevalue}$$

So, it is assumed that only if the argument is properly classified can $Y(W^T * X + b)$ be positive.

Loss function:

$$\text{Max}(0, 1 - Z_i)$$

Now, there are two cases, i. $Z \geq 1$ and ii. $Z < 1$.

If $Z \geq 1$, then $\text{Max}(0, 1 - Z) = 0$.

It means the point has been correctly classified and therefore the loss is 0.

If $Z < 1$, then $\text{Max}(0, 1 - Z) = 1 - Z$.

Final step:

$$Y(W^T * X + b) = 1 - \zeta$$

$$1 - Y(W^T * X + b) = \zeta$$

$$1 - Z = \zeta$$

Therefore, Z is the term that needs to be minimized.

Primal form of SVM:

$$\text{Argmin}\left(\frac{\|W\|}{2}\right) + C * \frac{1}{n} * \sum_{(i=1)}^n \zeta_i \quad (4.56)$$

Dual form of SVM:

Dual form of SVM is used to leverage the power of kernels which is a key feature of SVM.

$$\text{Max} \sum_{(i=1)}^n \alpha_i - \frac{1}{2} \sum_{(i=1)}^n \sum_{(j=1)}^n \alpha_i \alpha_j \gamma_i \gamma_j X_i^T X_j \quad (4.57)$$

Here, $\alpha \neq 0$ and equation (8) is equivalent to equation (9).

Final equation:

$$\text{Max} \sum_{(i=1)}^n \alpha_i - \frac{1}{2} \sum_{(i=1)}^n \sum_{(j=1)}^n \alpha_i \alpha_j \gamma_i \gamma_j K(X_i, X_j) \quad (4.58)$$

Kernel & Its Types:

There are several types of kernels. Two most popular among them are:

- i. Polynomial Kernel
- ii. Radial Basis Function or RBF Kernel

nu-SVM:

nu is a hyper-parameter that can be used to define the acceptable percentage error where $0 \leq nu \leq 1$. The parameter nu is the upper limit of the margin error fraction and the lower bound of the support vector fraction proportional to the total number of data-set training cases.

With nu hyper-parameter, two things can be done:

- i. It can control the percentage of error of a model.
- ii. It is unable to control but determines the number of support vectors.

4.8 Adam Optimizer

A stochastic gradient descent extension that can be used as a substitute to update iterative network weights based on training data for deep learning model training is called Adam optimizer (Brownlee, J. 2017)[23]. This algorithm can manage sparse gradients on a noisy topic and is also easy to customize and the default configuration parameters which perform well in problem solving most of the time. It works with momentum, as a combination of RMSprop and stochastic gradient descent. To adjust the learning rate Adam optimizer uses the squared gradients like RMSprop, hence it gets the advantage of momentum because it uses moving average like SGD.

4.9 K-Fold Cross Validation

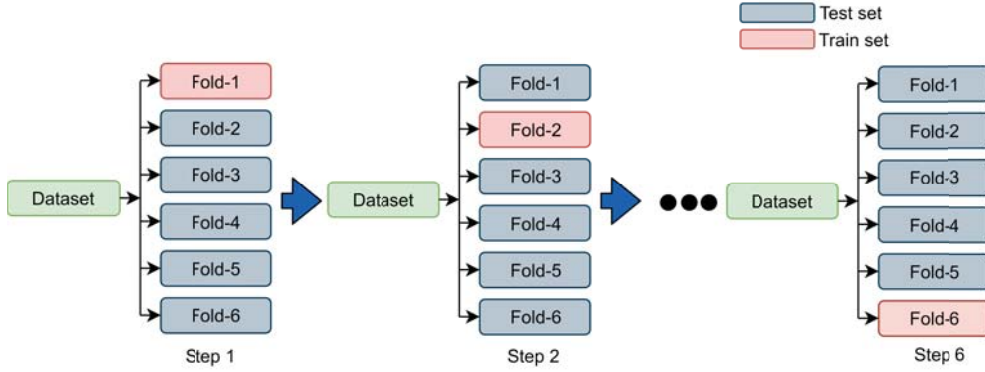


Figure 4.8: 6-fold cross validation

To avoid over fitting we have used k-fold CV(cross validation) where the data set is divided into k smaller sets, and then we trained a trained a model using $k - 1$ of the sets as train data where the remaining set is used as test set.[25]. The iteration is continued for k times where each time the testing set has been changed. Finally, the average performance of each step is considered as the final performance.

In our case, we have used 6-fold CV. This means that in each iteration, 5 data sets have been used for training and remaining 1 data set for testing.

4.10 F-score

”The F-score is the harmonic mean of precision (P) and recall (R)” [8].

$$\begin{aligned}
 F_1 &= \frac{2}{P^{-1} + R^{-1}} \\
 &= 2 \frac{P.R}{P + R}
 \end{aligned} \tag{1}$$

Precision or Confidence is the proportion of relevant instances among all retrieved instances[22].

$$Precision = tpa = \frac{tp}{tp + fp}$$

Recall or Sensitivity is the proportion of retrieved instances among all relevant instances[22].

$$Recall = tpr = \frac{tp}{tp + fn}$$

The general equation of F-score is,

$$F_{\beta} = \frac{(1 + \beta^2)PR}{\beta^2P + R} \quad (0 \leq \beta \leq +\infty) \quad (2)$$

In equation (2) β is a positive factor. which maintain balance between precision and recall. when $\beta = 1$, F-score also called as F_1 score (1) where P and R has same priority. If $\beta > 1$, R is more prioritized and if $\beta < 1$, P is more prioritized. In our case, we assume that, a crash which is undetected but occurred is harmful than a crash which is predicted but not occurred. Hence, recall gets more weight than precision in our case.

tpr = true positive Rate
 tp = true positive fp = false positive fn = false negative

Chapter 5

Result Analysis

Here, the test results are based on dataset of S&P 500 index, because the maximum number of crashes since 1950 has been reflected in the S&P 500 market index. Among the six datasets, five has been used to train and validation and remaining one (S&P 500) has been used for testing.

5.1 Log Periodic Power Law

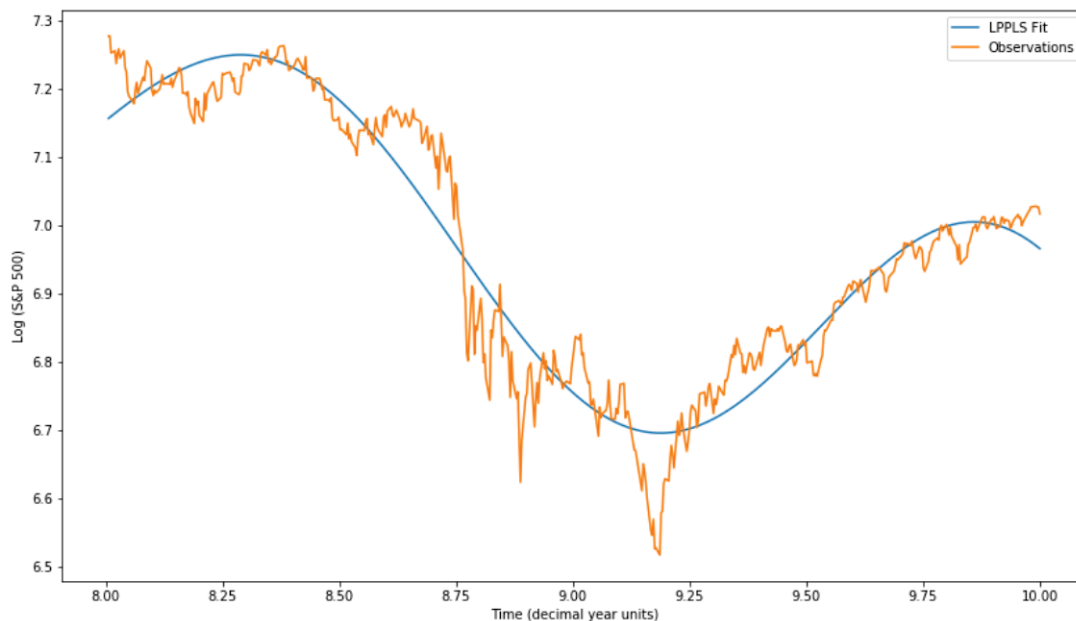


Figure 5.1: LPPL fit on S&P 500 market index from year of 2008 up to 2010

Log periodic power law is a highly non-linear function hence it is tricky to determine the parameters. Here the above LPPL-fitted diagram captures the major crash of 2008.

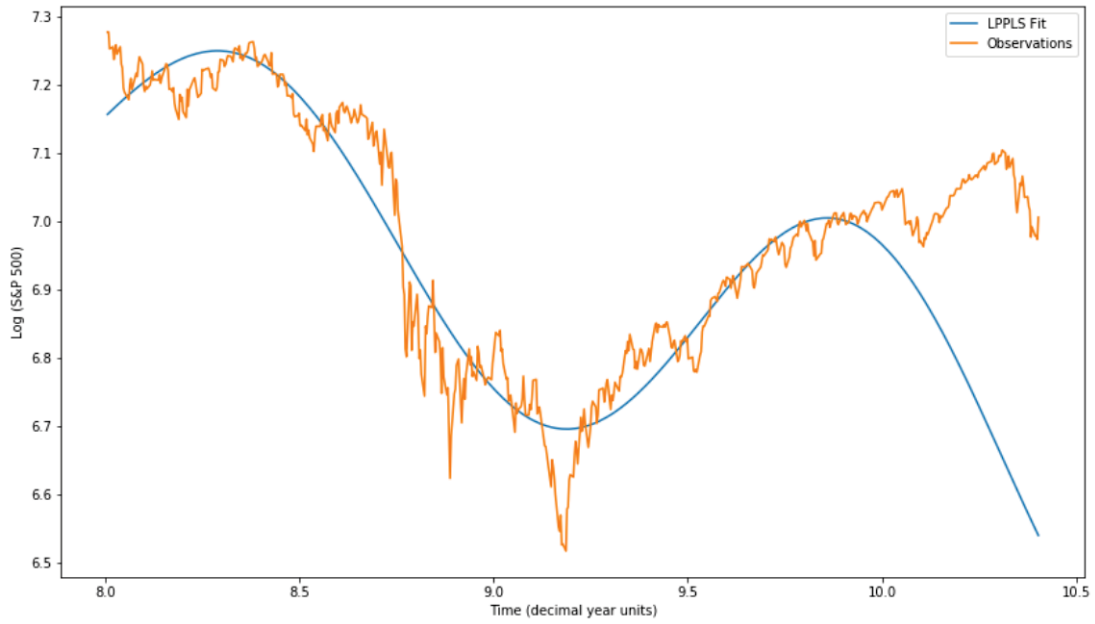


Figure 5.2: : LPPL prediction on S&P 500 market index from year of 2008 to 2010

For the COVID-19 market crash in year 2020:

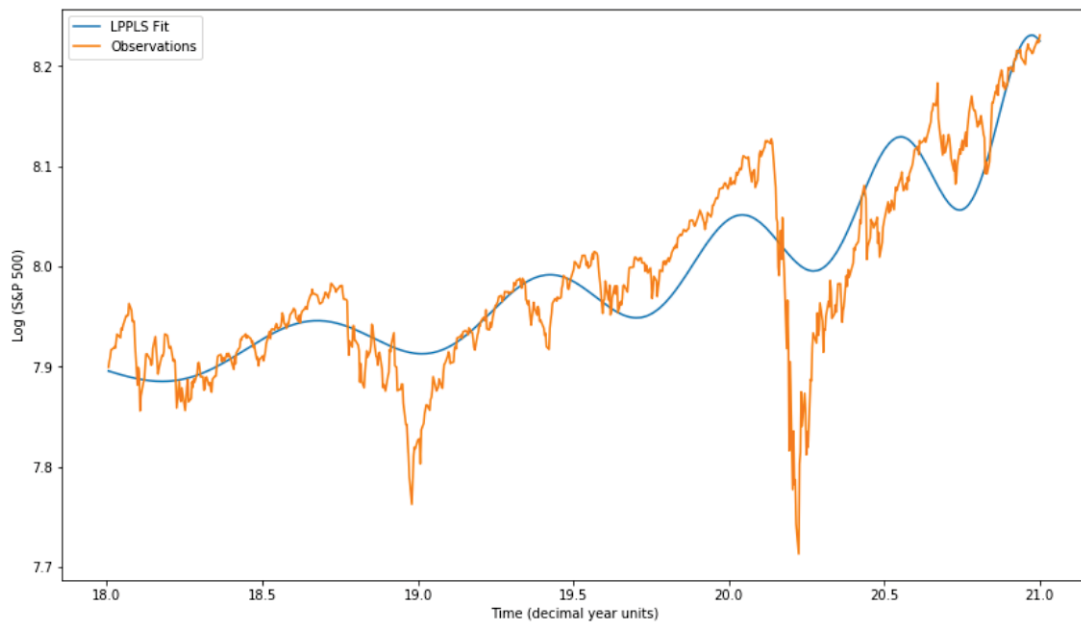


Figure 5.3: LPPL fit on S&P 500 market index from year of 2018 up to 2021

5.2 RNN LSTM

5.2.1 Stateful & Stateless LSTM

The recurrent neural networks can learn a sequence of data that other machine learning algorithms can not. However, when the sequence is

too large they might not be able to learn anything because of long-term dependencies. To prevent that problem, LSTM has been introduced where the entire cell state also passes into the next cell along with output.

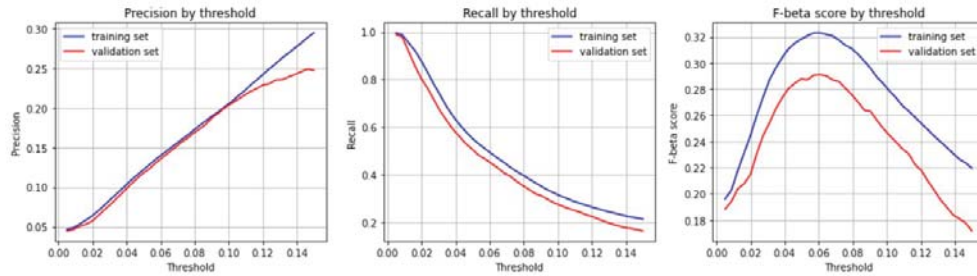


Figure 5.4: Finding the most accurate threshold for the test data set. (LSTM Stateless)

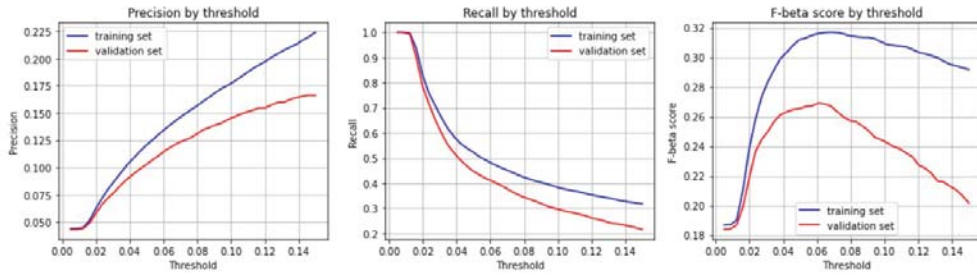


Figure 5.5: Finding the best best threshold for the test data set. (LSTM Stateful)

From the figure 5.14 and 5.15 we have decided to took threshold 0.07 for both stateless and stateful LSTM. Although this is not the exact threshold. We have tested with multiple threshold near to 0.07 and found the best recall value for the selected threshold. This threshold is strongly related with the test data set. If the data set get change, threshold might change also.

```

RNN LSTM

Predict crash in:          3 months
Threshold for positives:  0.07
Number of features:       10
Number of rows in training set: 47580
Number of epochs:         40
Sequence length:         5
Number of neurons/layer: 50
Batch size:               60
Optimizer:                adam
Loss function:            binary_crossentropy

```

```

Results for each train/val split:
      N225  SSE  HSI  BSESN  SMI  BVSP
positive actual train  0.04  0.04  0.04  0.04  0.04  0.04
positive pred train   0.13  0.12  0.13  0.13  0.13  0.19
precision train       0.17  0.16  0.15  0.15  0.14  0.11
recall train          0.52  0.41  0.44  0.45  0.42  0.47
accuracy_train        0.87  0.87  0.86  0.87  0.87  0.81
score_fbeta_train     0.36  0.31  0.32  0.32  0.30  0.29
positive actual val   0.05  0.04  0.04  0.04  0.05  0.04
positive pred val     0.16  0.12  0.12  0.12  0.12  0.14
precision val         0.09  0.14  0.11  0.11  0.16  0.14
recall val            0.31  0.47  0.30  0.31  0.37  0.51
accuracy_val          0.82  0.87  0.87  0.86  0.86  0.86
score_fbeta_val       0.20  0.32  0.22  0.22  0.29  0.33

```

```

Results - average over all train/val splits:
Positive train cases actual: 0.04
Positive train cases predicted: 0.14
Avg precision train (model/random): 0.15 / 0.04
Avg recall train (model/random): 0.45 / 0.14
Avg accuracy train (model/random): 0.86 / 0.83
Score train fbeta: 0.32 / 0.1
Positive validation cases actual: 0.04
Positive validation cases predicted: 0.14
Avg precision validation (model/random): 0.12 / 0.04
Avg recall validation (model/random): 0.38 / 0.14
Avg accuracy validation (model/random): 0.86 / 0.83
Score validation fbeta: 0.26 / 0.1

```

Figure 5.6: LSTM - Stateful

```

RNN LSTM

Predict crash in:          3 months
Threshold for positives:  0.07
Number of features:       28
Number of rows in training set: 46080
Number of epochs:         10
Sequence length:         10
Number of neurons/layer: 50
Batch size:               60
Optimizer:                adam
Loss function:            binary_crossentropy

```

```

Results for each train/val split:
      N225  SSE  HSI  BSESN  SMI  BVSP
positive actual train  0.04  0.05  0.05  0.04  0.04  0.04
positive pred train   0.16  0.18  0.12  0.11  0.09  0.10
precision train       0.16  0.13  0.16  0.17  0.17  0.16
recall_train          0.60  0.50  0.42  0.42  0.36  0.36
accuracy_train        0.85  0.82  0.87  0.89  0.90  0.89
score_fbeta_train     0.39  0.31  0.32  0.33  0.29  0.29
positive actual val   0.05  0.03  0.04  0.04  0.06  0.04
positive pred val     0.20  0.18  0.12  0.10  0.09  0.07
precision val         0.07  0.11  0.12  0.12  0.21  0.28
recall_val            0.31  0.62  0.38  0.29  0.32  0.49
accuracy_val          0.78  0.83  0.87  0.88  0.89  0.93
score_fbeta_val       0.19  0.33  0.26  0.23  0.29  0.43

```

```

Results - average over all train/val splits:
Positive train cases actual: 0.04
Positive train cases predicted: 0.13
Avg precision train (model/random): 0.16 / 0.04
Avg recall train (model/random): 0.44 / 0.13
Avg accuracy train (model/random): 0.87 / 0.84
Score train fbeta: 0.32 / 0.09
Positive validation cases actual: 0.04
Positive validation cases predicted: 0.14
Avg precision validation (model/random): 0.15 / 0.04
Avg recall validation (model/random): 0.4 / 0.14
Avg accuracy validation (model/random): 0.86 / 0.83
Score validation fbeta: 0.29 / 0.1

```

Figure 5.7: LSTM - Stateless

Test results:

```

Test results (test set: S&P 500):
Positive test cases actual: 0.04
Positive test cases predicted: 0.12
Precision test (model/random): 0.13 / 0.04
Recall test (model/random): 0.38 / 0.12
Accuracy test (model/random): 0.87 / 0.85
Score test fbeta: 0.28 / 0.09

```

Figure 5.8: Test result: Stateful

```

Test results (test set: S&P 500):
Positive test cases actual: 0.04
Positive test cases predicted: 0.17
Precision test (model/random): 0.14 / 0.04
Recall test (model/random): 0.53 / 0.17
Accuracy test (model/random): 0.84 / 0.81
Score test fbeta: 0.34 / 0.1

```

Figure 5.9: Test result: Stateless

Statefull LSTM has slightly higher accuracy than the stateless LSTM.

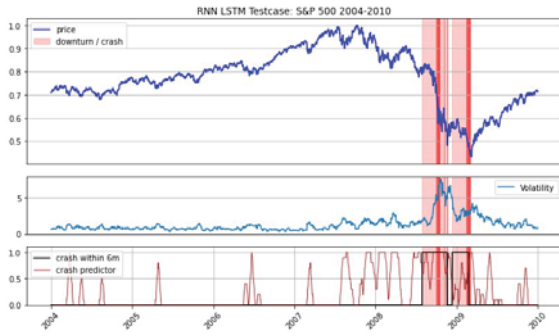


Figure 5.10: LSTM stateful test case 2004 to 2016

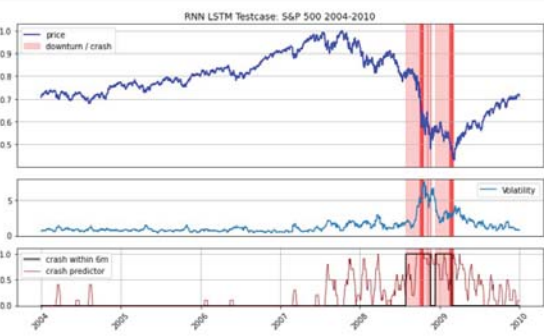


Figure 5.11: LSTM stateless test case 2004 to 2016

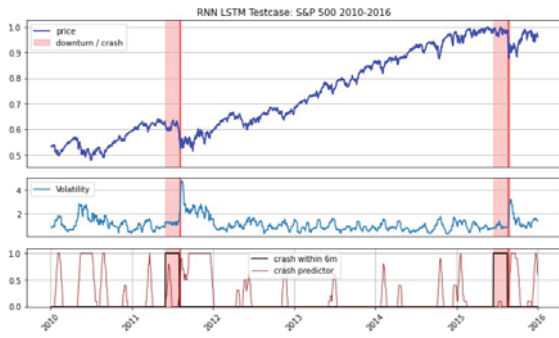


Figure 5.12: LSTM stateful test case during Covid-19



Figure 5.13: LSTM stateless test case during Covid-19

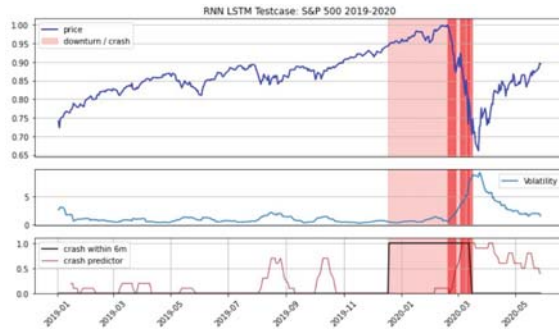


Figure 5.12: LSTM stateful test case during Covid-19

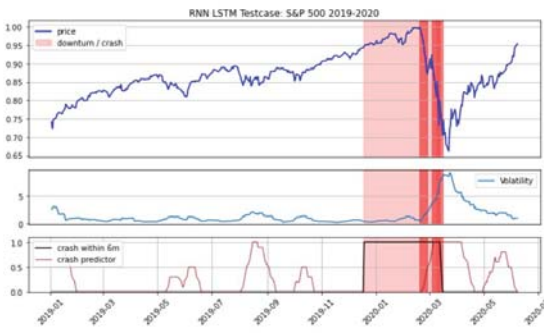
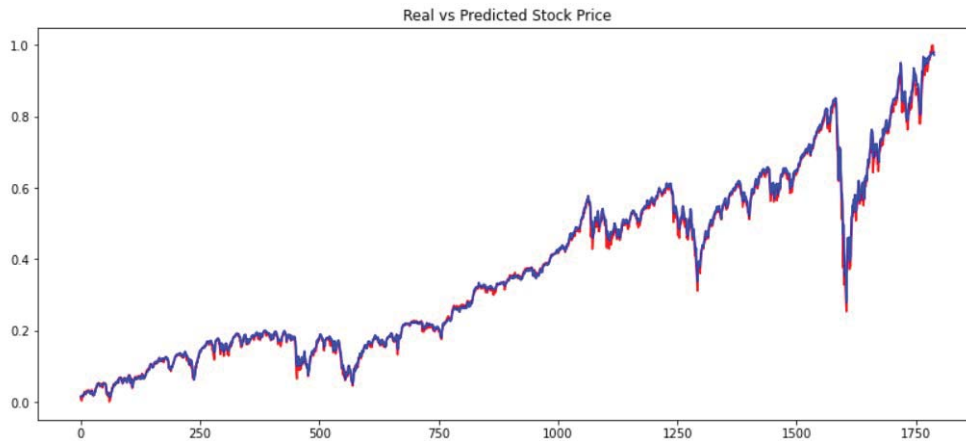


Figure 5.13: LSTM stateless test case during Covid-19

5.3 Back Propagation



Number of layer layer: 5 (including input and output layer) Activation function: Relu, Sigmoid. Optimizer: Adam Epoch: 200 Loss in first Epoch: $9.6601e-04$ Loss in last Epoch: $3.0428e-04$ Mean squared error: 67.62911121142531

5.4 Regression

5.4.1 Linear & Logistic

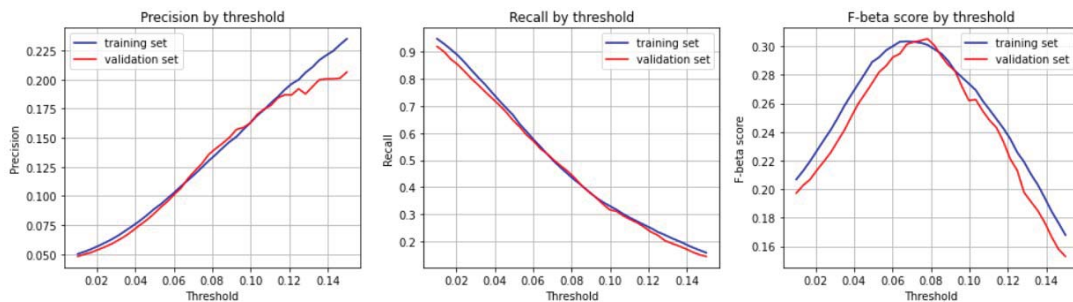


Figure 5.14: Finding the best best threshold for the test data set.(Linear regression)

Here in the regression case, the best beta score we have found is 0.075.

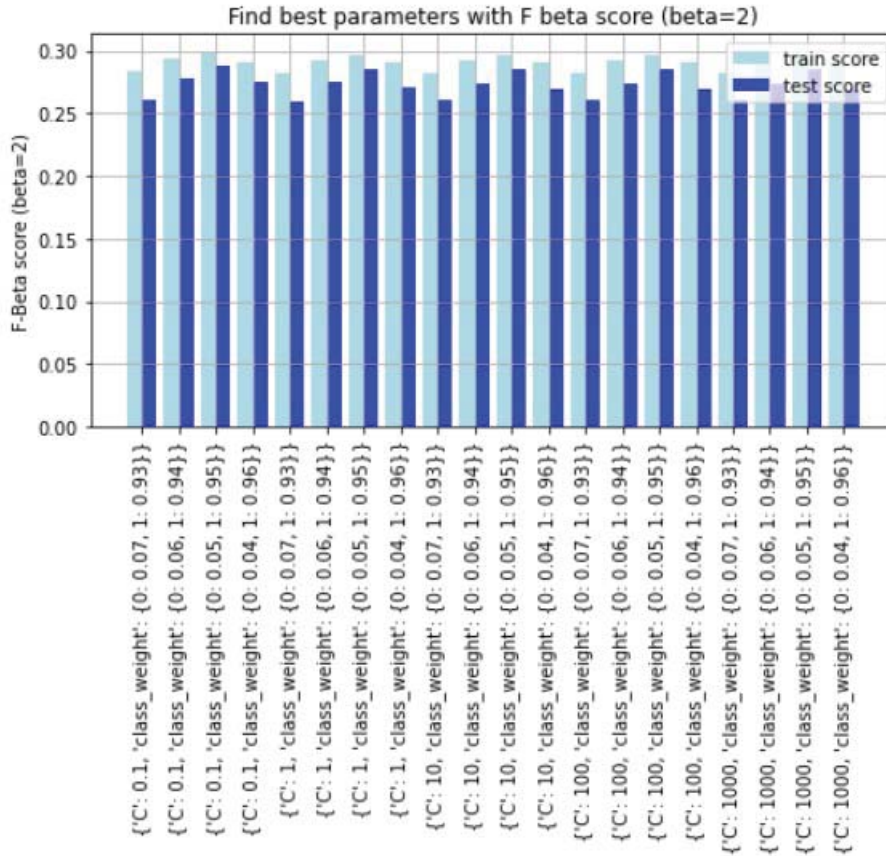


Figure 5.15: Finding the best best threshold for the test data set. (Logistic regression)

Linear Regression

Predict crash in: 3 months
 Threshold for positives: 0.075
 Number of features: 16
 Number of rows for training: 46029

Results for each train/val split:

	N225	SSE	HSI	BSESN	SMI	BVSP
positive actual train	0.04	0.05	0.05	0.04	0.04	0.04
positive pred train	0.17	0.16	0.17	0.16	0.15	0.16
precision train	0.14	0.12	0.12	0.13	0.12	0.12
recall train	0.56	0.44	0.47	0.47	0.43	0.43
accuracy_train	0.84	0.83	0.82	0.83	0.84	0.84
score_fbeta train	0.35	0.29	0.30	0.30	0.28	0.29
positive actual val	0.05	0.03	0.04	0.04	0.06	0.04
positive pred val	0.22	0.16	0.17	0.15	0.13	0.12
precision val	0.07	0.15	0.11	0.11	0.16	0.18
recall val	0.33	0.70	0.50	0.37	0.39	0.56
accuracy val	0.76	0.86	0.83	0.83	0.85	0.88
score fbeta val	0.19	0.40	0.29	0.25	0.30	0.40

Results - average over all train/val splits:

Positive train cases actual: 0.04
 Positive train cases predicted: 0.16
 Avg precision train (model/random): 0.13 / 0.04
 Avg recall train (model/random): 0.47 / 0.16
 Avg accuracy train (model/random): 0.83 / 0.81
 Score train fbeta: 0.3 / 0.11
 Positive validation cases actual: 0.04
 Positive validation cases predicted: 0.17
 Avg precision validation (model/random): 0.13 / 0.04
 Avg recall validation (model/random): 0.48 / 0.17
 Avg accuracy validation (model/random): 0.84 / 0.8
 Score validation fbeta: 0.3 / 0.11

Figure 5.16: Lin. Regression

Logistic Regression

Predict crash in: 3 months
 Number of features: 16
 Number of rows in training set: 46029

Results for each train/val split:

	N225	SSE	HSI	BSESN	SMI	BVSP
positive actual train	0.04	0.05	0.05	0.04	0.04	0.04
positive pred train	0.14	0.16	0.17	0.15	0.15	0.16
precision train	0.16	0.12	0.12	0.13	0.12	0.12
recall train	0.53	0.44	0.45	0.44	0.41	0.42
accuracy_train	0.87	0.83	0.83	0.84	0.84	0.84
score_fbeta train	0.36	0.29	0.29	0.29	0.27	0.28
positive actual val	0.05	0.03	0.04	0.04	0.06	0.04
positive pred val	0.18	0.16	0.17	0.15	0.13	0.13
precision val	0.07	0.15	0.11	0.10	0.16	0.18
recall val	0.26	0.70	0.49	0.32	0.37	0.56
accuracy val	0.80	0.86	0.83	0.84	0.86	0.88
score fbeta val	0.17	0.40	0.28	0.22	0.29	0.39

Results - average over all train/val splits:

Positive train cases actual: 0.04
 Positive train cases predicted: 0.15
 Avg precision train (model/random): 0.13 / 0.04
 Avg recall train (model/random): 0.45 / 0.15
 Avg accuracy train (model/random): 0.84 / 0.82
 Score train fbeta: 0.3 / 0.1
 Positive validation cases actual: 0.04
 Positive validation cases predicted: 0.16
 Avg precision validation (model/random): 0.13 / 0.04
 Avg recall validation (model/random): 0.45 / 0.16
 Avg accuracy validation (model/random): 0.84 / 0.81
 Score validation fbeta: 0.29 / 0.1

Figure 5.17: Log. Regression

Test results:

Test results (test set: S&P 500):
 Positive test cases actual: 0.04
 Positive test cases predicted: 0.16
 Precision test (model/random): 0.15 / 0.04
 Recall test (model/random): 0.54 / 0.16
 Accuracy test (model/random): 0.85 / 0.81
 Score test fbeta: 0.35 / 0.1

Test results (test set: S&P 500):
 Positive test cases actual: 0.04
 Positive test cases predicted: 0.15
 Precision test (model/random): 0.15 / 0.04
 Recall test (model/random): 0.53 / 0.15
 Accuracy test (model/random): 0.85 / 0.82
 Score test fbeta: 0.35 / 0.1

Figure 5.18: Test result: Linear

Figure 5.19: Test result: Logistic

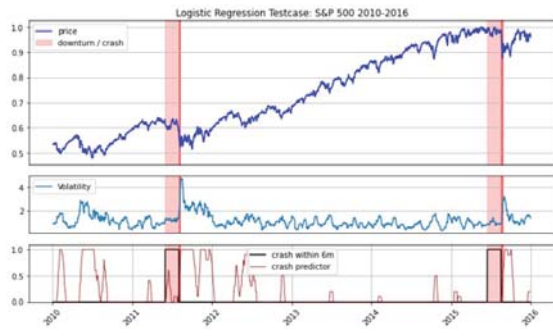
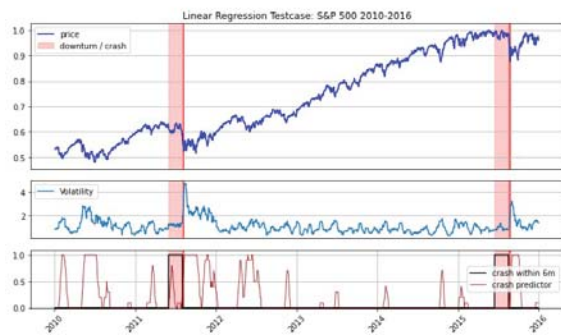
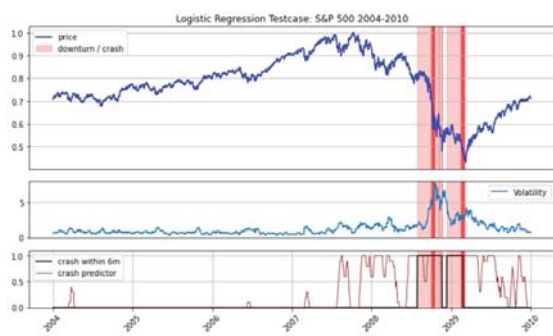
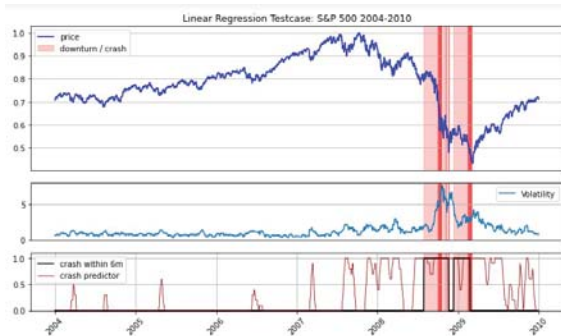


Figure 5.20: Lin. regression test case 2004 to 2016

Figure 5.21: Log. regression test case 2004 to 2016

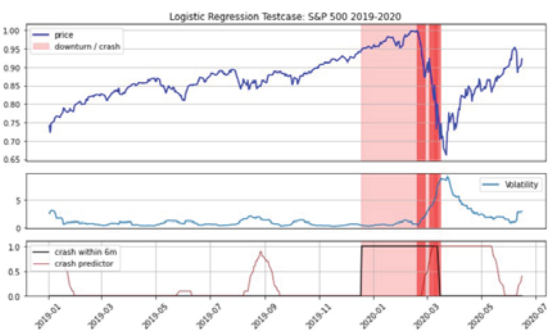
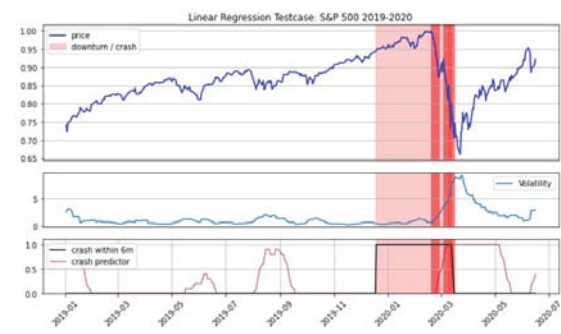


Figure 5.22: Lin. Regression test case during Covid-19

Figure 5.23: Log. Regression test case during Covid-19

Prediction results:

Linear Regression prediction of a crash within 1 months: 0.02
Linear Regression prediction of a crash within 3 months: 0.52
Linear Regression prediction of a crash within 6 months: 0.96

Logistic Regression prediction of a crash within 1 months: 0.0
Logistic Regression prediction of a crash within 3 months: 0.3
Logistic Regression prediction of a crash within 6 months: 1.0

Figure 5.24: Prediction: Linear

Figure 5.25: Prediction: Logistic

5.5 Decision Tree

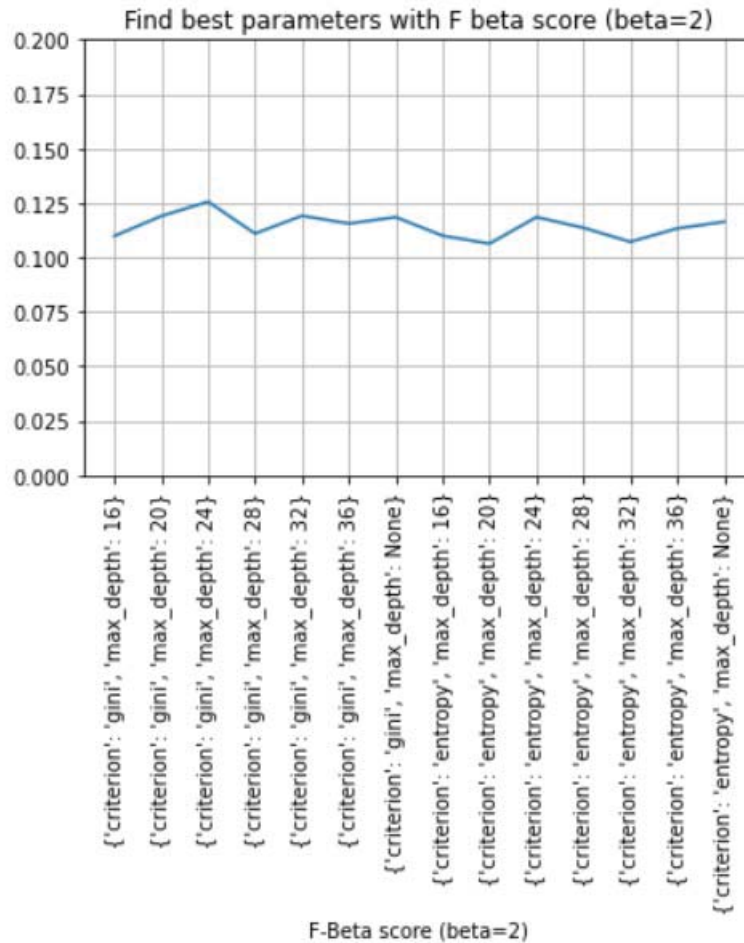


Figure 5.26: Finding best parameters

Among all the machine learning model we have applied, the performance of decision tree was very poor compared to other models. We have got average recall value of 0.12 for the validation set and 0.07 for the testing set.

Decision Trees

Predict crash in: 3 months
Threshold for positives: None
Number of features: 16
Number of rows for training: 46029

Results for each train/val split:

	N225	SSE	HSI	BSESN	SMI	BVSP
positive actual train	0.04	0.05	0.05	0.04	0.04	0.04
positive pred train	0.04	0.05	0.05	0.04	0.04	0.04
precision train	1.00	1.00	1.00	1.00	1.00	1.00
recall train	1.00	1.00	1.00	1.00	1.00	1.00
accuracy_train	1.00	1.00	1.00	1.00	1.00	1.00
score_fbeta train	1.00	1.00	1.00	1.00	1.00	1.00
positive actual val	0.05	0.03	0.04	0.04	0.06	0.04
positive pred val	0.04	0.05	0.03	0.06	0.04	0.03
precision val	0.08	0.11	0.12	0.09	0.12	0.12
recall val	0.08	0.19	0.12	0.12	0.09	0.11
accuracy val	0.92	0.92	0.94	0.91	0.91	0.94
score_fbeta val	0.08	0.17	0.12	0.11	0.09	0.11

Results - average over all train/val splits:

Positive train cases actual: 0.04
Positive train cases predicted: 0.04
Avg precision train (model/random): 1.0 / 0.04
Avg recall train (model/random): 1.0 / 0.04
Avg accuracy train (model/random): 1.0 / 0.92
Score train fbeta: 1.0 / 0.04
Positive validation cases actual: 0.04
Positive validation cases predicted: 0.04
Avg precision validation (model/random): 0.11 / 0.04
Avg recall validation (model/random): 0.12 / 0.04
Avg accuracy validation (model/random): 0.92 / 0.92
Score validation fbeta: 0.11 / 0.04

Figure 5.27: Decision tree model overview

Test result:

```
Test results (test set: S&P 500):  
Positive test cases actual: 0.04  
Positive test cases predicted: 0.04  
Precision test (model/random): 0.09 / 0.04  
Recall test (model/random): 0.07 / 0.04  
Accuracy test (model/random): 0.93 / 0.93  
Score test fbeta: 0.08 / 0.04
```

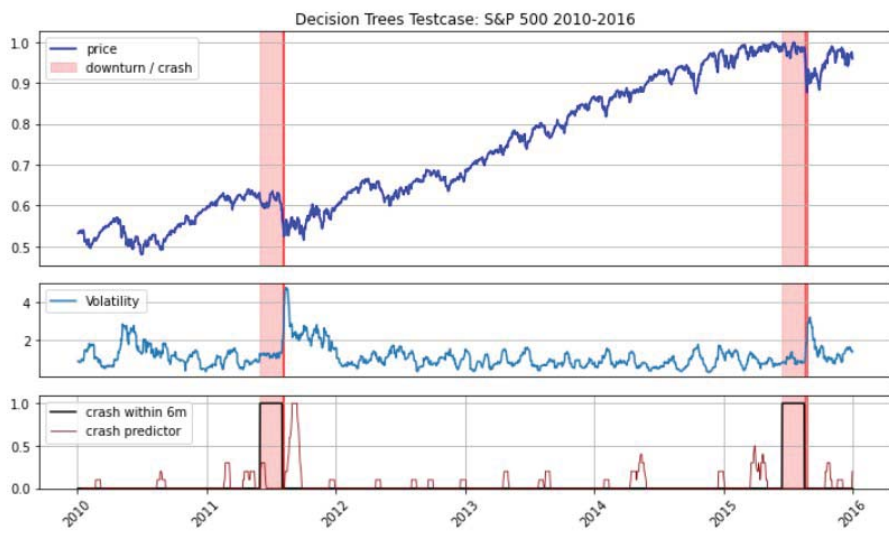
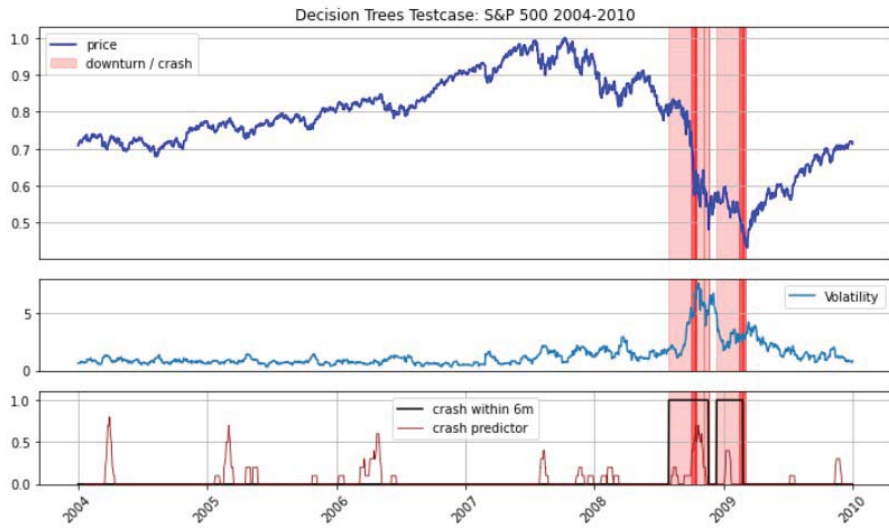


Figure 5.28: Decision Tree test case from 2004 to 2016

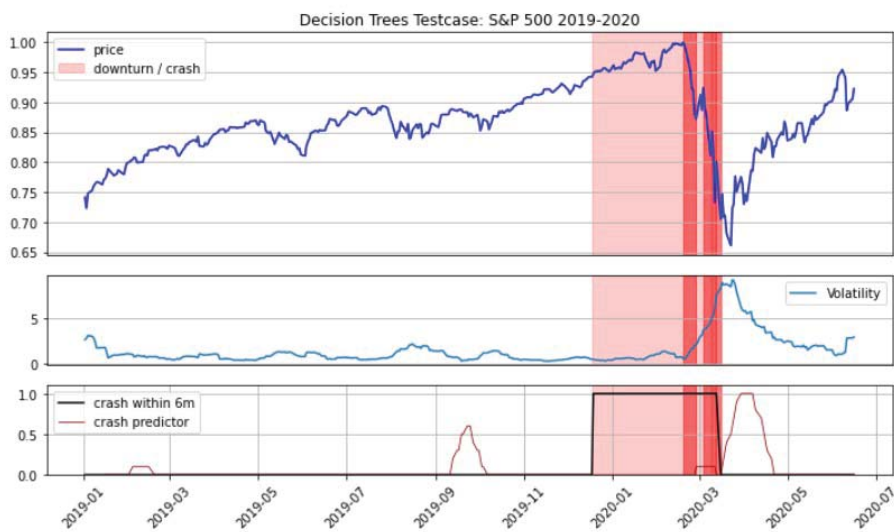


Figure 5.29: Decision Tree test case during Covid-19

5.6 Support Vector Machine

```
SVM: linear Classification

Predict crash in:          3 months
Threshold for positives:  None
Number of features:       16
Number of rows in training set: 46029

Results for each train/val split:
      N225  SSE  HSI  BSESN  SMI  BVSP
positive actual train  0.04  0.05  0.05  0.04  0.04  0.04
positive pred train    0.13  0.13  0.14  0.13  0.12  0.12
precision train        0.17  0.13  0.13  0.14  0.13  0.13
recall train           0.51  0.39  0.41  0.41  0.38  0.37
accuracy_train         0.87  0.86  0.85  0.86  0.87  0.87
score_fbeta_train      0.36  0.28  0.29  0.30  0.27  0.27
positive actual val    0.05  0.03  0.04  0.04  0.06  0.04
positive pred val      0.17  0.13  0.14  0.12  0.11  0.09
precision val          0.07  0.18  0.11  0.11  0.18  0.21
recall val             0.24  0.70  0.42  0.30  0.35  0.48
accuracy_val           0.80  0.89  0.86  0.86  0.87  0.91
score_fbeta_val        0.16  0.44  0.27  0.22  0.29  0.38

Results - average over all train/val splits:
Positive train cases actual:      0.04
Positive train cases predicted:    0.13
Avg precision train (model/random): 0.14 / 0.04
Avg recall train (model/random):    0.41 / 0.13
Avg accuracy train (model/random):  0.86 / 0.84
Score train fbeta:                 0.3 / 0.09
Positive validation cases actual:  0.04
Positive validation cases predicted: 0.13
Avg precision validation (model/random): 0.14 / 0.04
Avg recall validation (model/random):  0.42 / 0.13
Avg accuracy validation (model/random): 0.87 / 0.83
Score validation fbeta:             0.3 / 0.09
```

Figure 5.30: Support vecto machine model overview

Test result:

```
Test results (test set: S&P 500):
Positive test cases actual:      0.04
Positive test cases predicted:    0.12
Precision test (model/random):   0.16 / 0.04
Recall test (model/random):      0.47 / 0.12
Accuracy test (model/random):    0.87 / 0.84
Score test fbeta:                0.34 / 0.09
```

Figure 5.31: Test result of SVM on S&P 500 data set

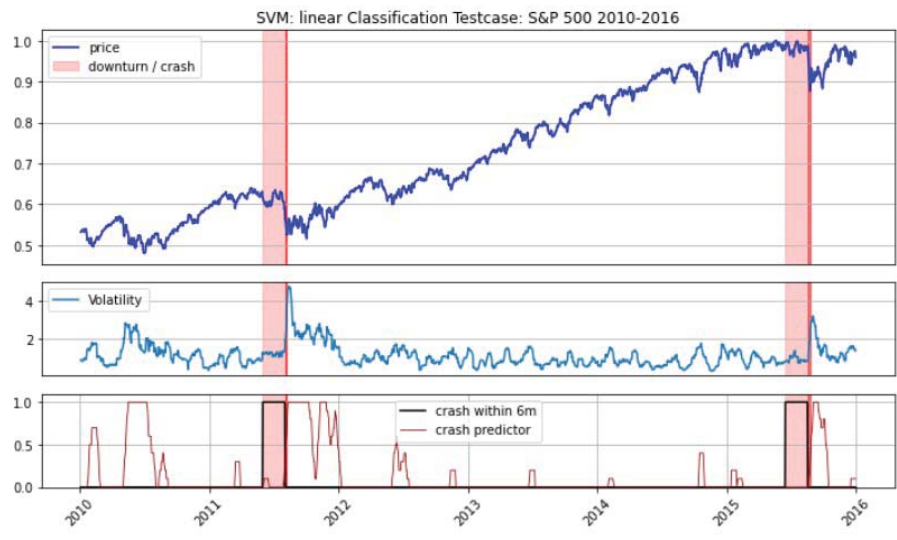
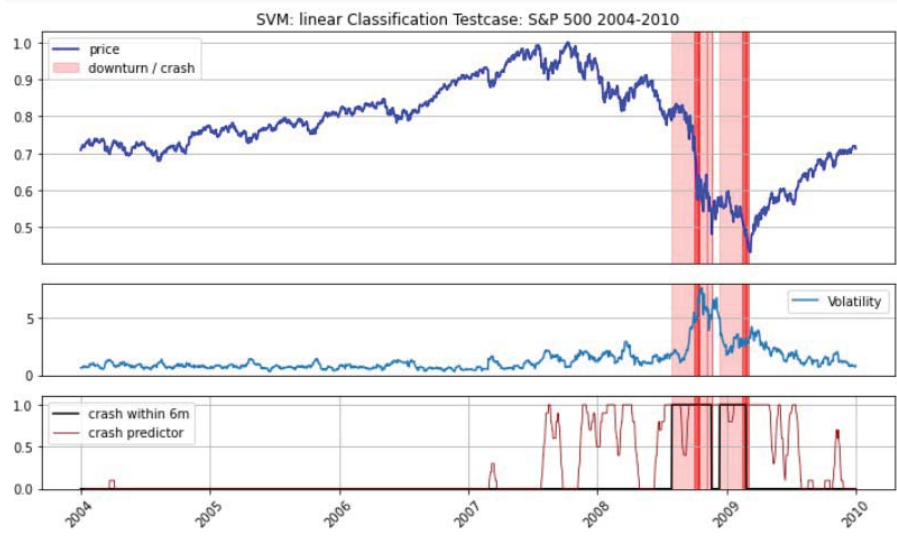


Figure 5.32: SVM from the year 2004 to year 2016

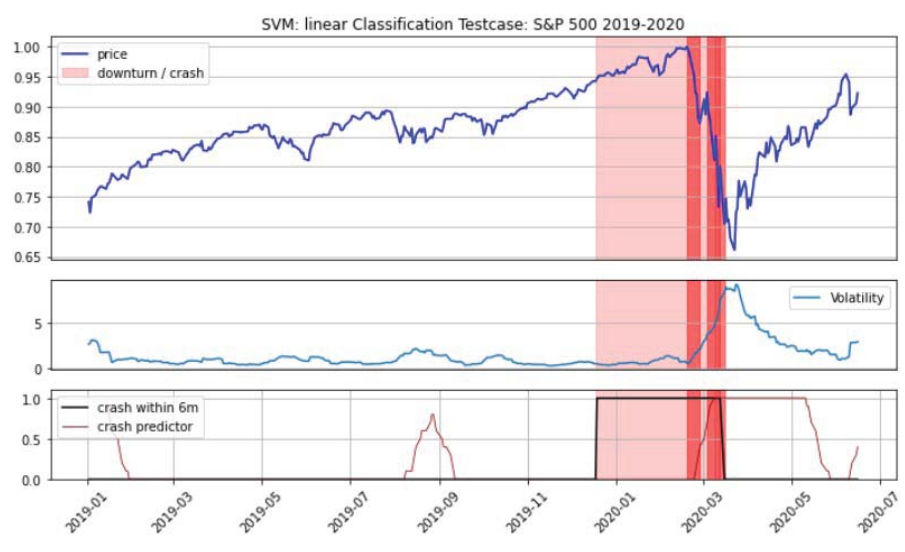


Figure 5.33: SVM test case during Covid-19

Predicted result:

```
SVM: linear Classification prediction of a crash within 1 months: 0.0  
SVM: linear Classification prediction of a crash within 3 months: 0.03  
SVM: linear Classification prediction of a crash within 6 months: 1.0
```

5.7 Result Overview

From the above analysis, we have found that, among the regression, SVM and decision tree, the best score obtained from SVM models is almost similar to the results obtained from regression models. This makes regression models more preferable because regression models can be trained much faster than the SVM model. The decision trees have not been able to produce satisfactory result as any of the other tested models. The RNN with LSTM is capable of learn complex price structure which regression models are not. In practice, tuning the parameter fro RNN is much difficult compared to others. The best score we have got with a sequential layer followed two LSTM layers.

Chapter 6

Conclusion and Motivation for Future work

In brief, in this research, we have focused on the extreme events of stock markets, meaning financial crashes and have used Neural Networks as a new means to determine speculative bubbles. Specifically, we have described the stock market behaviour before and after a crash. We have also exploited the positive Black Swan Events while handling the negative after-effects of stock market crashes. We have studied Machine Learning and Neural Networks and used the Risk-Driven Model to attain our goal. In order to make our project fruitful, we have studied the history of several previous stock market crashes and also the similarities and differences among them. Besides, we have researched deep learning techniques regarding every aspect of the problem. We have fit crash data as per our target with the knowledge we have attained using the mathematical model of Log-Periodic Power Law. We have used best neural network techniques for stock price fitting which are back propagation and support vector machines. We have conducted analysis of data sets of large markets of the stock exchange and online financial data servers about various incidents of crash of mainly the U.S. financial markets and will be learning more about this aspect. We have implemented the back propagation algorithm of Neural Networks for the prediction of stock market crashes and severe changes of price in general and obtained robust results. Our results are still not perfect, as the price patterns of stock market are so very complex, but there are still a lot of scopes to work on the complex price patterns. So, in the near future we plan on continuing our work and research and also want to test and use several other models to achieve our goal.

Bibliography

- [1] D. Sornette and A. Johansen, “Large financial crashes”, *Physica A-statistical Mechanics and Its Applications*, vol. 245, pp. 411–422, 1997.
- [2] D. Sornette and A. Johansen, “Large financial crashes”, *Physica A: Statistical Mechanics and its Applications*, vol. 245, no. 3-4, pp. 411–422, 1997.
- [3] M. Buscema, “Back propagation neural networks”, *Substance use & misuse*, vol. 33, no. 2, pp. 233–270, 1998.
- [4] A. Johansen and D. Sornette, “Large stock market price drawdowns are outliers”, *The Journal of Risk*, vol. 4, no. 2, pp. 69–110, 2001. DOI: 10.21314/jor.2002.058.
- [5] D. Sornette, “Critical market crashes”, *Physics Reports*, vol. 378, no. 1, pp. 1–98, 2003. DOI: 10.1016/s0370-1573(02)00634-8.
- [6] V. Kecman, “Support vector machines—an introduction”, in *Support vector machines: theory and applications*, Springer, 2005, pp. 1–47.
- [7] W. S. Noble, “What is a support vector machine?”, *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.
- [8] Y. Sasaki, “The truth of the f-measure”, *Teach Tutor Mater*, Jan. 2007.
- [9] L. Ren and Y. U. Glasure, “Applicability of the revised mean absolute percentage errors (mape) approach to some popular normal and non-normal independent time series”, *International Advances in Economic Research*, vol. 15, pp. 409–420, 2009.
- [10] S. Haykin, “Neural networks and learning machines”, 2010.
- [11] M. Cilimkovic, “Neural networks and back propagation algorithm”, *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, vol. 15, 2015.
- [12] S. Patil, K. Patidar, and M. Jain, “A survey on stock market prediction using svm”, 2016.
- [13] D. Sornette, “Financial crashes are “outliers””, *Why Stock Markets Crash*, 2017. DOI: 10.23943/princeton/9780691175959.003.0003.
- [14] —, *Why stock markets crash: critical events in complex financial systems*. Princeton University Press, 2017, vol. 49.
- [15] N. N. Taleb, *The black swan: the impact of the highly improbable*. Taylor and Francis, 2017.
- [16] T.-W. Chien, J. C. Chow, Y. Chang, and W. Chou, “Applying gini coefficient to evaluate the author research domains associated with the ordering of author names: A bibliometric study”, *Medicine*, vol. 97, no. 39, 2018.

- [17] B. M. Henrique, V. A. Sobreiro, and H. Kimura, “Stock price prediction using support vector regression on daily and up to the minute prices”, *The Journal of finance and data science*, vol. 4, no. 3, pp. 183–201, 2018.
- [18] Y.-G. Song, Y.-L. Zhou, and R.-J. Han, “Neural networks for stock price prediction”, *arXiv preprint arXiv:1805.11317*, 2018.
- [19] G. Gasso, “Logistic regression”, 2019.
- [20] K. Amadeo, *When and why did the stock market crash in 2008?*, Apr. 2020. [Online]. Available: <https://www.thebalance.com/stock-market-crash-of-2008-3305535>.
- [21] R. Chowdhury, M. Mahdy, T. N. Alam, G. D. A. Quaderi, and M. A. Rahman, “Predicting the stock price of frontier markets using machine learning and modified black–scholes option pricing model”, *Physica A-statistical Mechanics and Its Applications*, vol. 555, p. 124 444, 2020.
- [22] D. Powers, “Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation”, *ArXiv*, vol. abs/2010.16061, 2020.
- [23] J. Brownlee, *Gentle introduction to the adam optimization algorithm for deep learning*, Jan. 2021. [Online]. Available: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20is%20an%20optimization%20algorithm,iterative%20based%20in%20training%20data.&text=The%20algorithm%20is%20called%20Adam,derived%20from%20adaptive%20moment%20estimation..>
- [24] *16 candlestick patterns every trader should know*. [Online]. Available: <https://www.ig.com/en/trading-strategies/16-candlestick-patterns-every-trader-should-know-180615>.
- [25] *3.1. cross-validation: Evaluating estimator performance*¶. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html.